Hiroyuki Waki et al.
S.N. 09/288,263

日本国特許庁
PATENT OFFICE
JAPANESE GOVERNMENT

NAKE-BG55

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。
This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: １９９８年 ４月 ８日

出願番号
Application Number: 平成１０年特許願第０９６２０４号

出願人
Applicant (s): 松下電器産業株式会社

CERTIFIED COPY OF
PRIORITY DOCUMENT

１９９９年 ３月 ５日

特許庁長官
Commissioner,
Patent Office 伊佐山建志

出証番号 出証特平１１－３０１１８５１

【書類名】　特許願

【整理番号】　2022590401

【提出日】　平成10年 4月 8日

【あて先】　特許庁長官　荒　井　寿　光　殿

【国際特許分類】　G06F　5/00

【発明の名称】　仮想マシン及びコンパイラ

【請求項の数】　30

【発明者】

【住所又は居所】　大阪府門真市大字門真１００６番地　松下電器産業株式会社内

【氏名】　和氣　裕之

【発明者】

【住所又は居所】　大阪府門真市大字門真１００６番地　松下電器産業株式会社内

【氏名】　井上　信治

【発明者】

【住所又は居所】　大阪府門真市大字門真１００６番地　松下電器産業株式会社内

【氏名】　葉山　悟

【発明者】

【住所又は居所】　大阪府門真市大字門真１００６番地　松下電器産業株式会社内

【氏名】　藤田　光子

【発明者】

【住所又は居所】　大阪府門真市大字門真１００６番地　松下電器産業株式会社内

【氏名】　石川　亮

【特許出願人】

【識別番号】　000005821

【氏名又は名称】　松下電器産業株式会社

【代理人】

【識別番号】　100090446

【弁理士】

【氏名又は名称】　中島　司朗

【手数料の表示】

【予納台帳番号】　014823

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9003742

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　仮想マシン及びコンパイラ

【特許請求の範囲】

【請求項１】

実マシンによる制御の下で仮想マシン命令を実行するスタック型仮想マシンであって、

データを後入れ先出し方式で一時的に記憶するためのスタック手段と、

実行対象となる仮想マシン命令列と、それら仮想マシン命令それぞれに対応づけられた情報であって、対応づけられた仮想マシン命令に後続する仮想マシン命令が実行された場合の前記スタック手段でのデータの格納状態の変化を示す先行命令情報とを記憶する命令記憶手段と、

前記命令記憶手段から次に実行すべき仮想マシン命令及び対応する先行命令情報を読み出す読み出し手段と、

読み出された仮想マシン命令と先行命令情報との組み合わせに対応する演算処理を特定し実行する解読実行手段と

を備えることを特徴とする仮想マシン。

【請求項２】

前記解読実行手段は、

対象とする全ての種類の仮想マシン命令と全ての種類の先行命令情報との組み合わせそれぞれに対応する実マシン命令列を記憶する実マシン命令列記憶部と、

前記読み出し手段によって読み出された仮想マシン命令と先行命令情報との組み合わせに対応する実マシン命令列を前記実マシン命令列記憶部に記憶された実マシン命令列から特定する特定部と、

特定された実マシン命令列を実行する実行部と

を有することを特徴とする請求項１記載の仮想マシン。

【請求項３】

前記先行命令情報は、対応する仮想マシン命令に後続する仮想マシン命令が実行された場合の前記スタック手段に格納されるデータ数の増減を示し、

前記実マシン命令列記憶部に記憶された実マシン命令列には、対応する先行命

令情報に基づいて前記スタック手段を先行処理する実マシン命令が含まれていることを特徴とする請求項２記載の仮想マシン。

【請求項４】

前記実マシン命令列記憶部に記憶された実マシン命令列は、前記スタック手段において最後に格納されたデータと最後から２番目に格納されたデータそれぞれを記憶する領域が前記実マシンが備える２つのレジスタに割り当てられていることを前提とする内容であることを特徴とする請求項３記載の仮想マシン。

【請求項５】

前記命令記憶手段は、前記仮想マシン命令列を記憶する第１領域と、その第１領域における各仮想マシン命令の記憶位置に関連付けられた記憶位置に前記先行命令情報を記憶する第２領域とからなり、

前記読み出し手段は、前記第１領域と前記第２領域それぞれの対応する記憶領域から前記仮想マシン命令と先行命令情報との組を読み出すことを特徴とする請求項１記載の仮想マシン。

【請求項６】

前記命令記憶手段は、前記仮想マシン命令列と前記先行命令情報とを、対応する１つの仮想マシン命令と１つの先行命令情報との組を１つの拡張仮想マシン命令とする拡張仮想マシン命令列として記憶し、

前記読み出し手段は、前記命令記憶手段から拡張仮想マシン命令を読み出し、

前記解読実行手段は、読み出された拡張仮想マシン命令に対応する演算処理を特定し実行することを特徴とする請求項１記載の仮想マシン。

【請求項７】

スタック型仮想マシンを対象とするコンパイラであって、

ソースプログラムを前記仮想マシンが実行する仮想マシン命令列に変換する命令列変換手段と、

変換によって得られた仮想マシン命令列を構成する仮想マシン命令それぞれについて、後続する仮想マシン命令が前記仮想マシンによって実行された場合の前記スタック手段でのデータの格納状態の変化を示す先行命令情報を生成する先行命令情報生成手段と、

生成された先行命令情報と対応する仮想マシン命令とを関連付けて出力する関連付け手段と

を備えることを特徴とする仮想マシンコンパイラ。

【請求項８】

実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、

実行対象となる仮想マシン命令列を記憶する命令記憶手段と、

前記命令記憶手段から次に実行すべき仮想マシン命令を読み出す読み出し手段と、

読み出された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、

前記解読実行手段は、

読み出された仮想マシン命令が実行制御の流れを分岐させる分岐命令であるか否かを判定する分岐命令判定部と、

分岐命令であると判定される度に、その分岐命令の実行に加えて、この仮想マシンへの割込み要求の発生の有無の検出と、発生している場合の割込み処理とを実行する割込み処理部とを有することを特徴とする仮想マシン。

【請求項９】

前記解読実行手段はさらに、

対象とする全ての種類の仮想マシン命令に対応する実マシン命令列とこの仮想マシンへの割込み要求に対処するための実マシン命令列を記憶する実マシン命令列記憶部と、

指定された実マシン命令列を実行する実行部とを備え、

前記割込み処理部は、前記分岐命令判定部によって分岐命令であると判定される度に、前記実行部に、前記割込み要求に対処するための実マシン命令列を実行させた後に前記分岐命令に対応する実マシン命令列を実行させることを特徴とする請求項８記載の仮想マシン。

【請求項１０】

実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、

実行対象となる仮想マシン命令列を記憶する命令記憶手段と、

前記命令記憶手段から次に実行すべき仮想マシン命令を読み出す読み出し手段と、

読み出された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、

前記解読実行手段は、

読み出された仮想マシン命令が一定個数の仮想マシン命令の集まりをブロックとした場合の各ブロックを代表する仮想マシン命令に相当するか否かを判定するブロック判定部を備え、

各ブロックを代表する仮想マシン命令であると判定される度に、その仮想マシン命令の実行に加えて、この仮想マシンへの割込み要求の発生の有無の検出と、発生している場合の割込み処理とを実行する割込み処理部とを有することを特徴とする仮想マシン。

【請求項１１】

前記解読実行手段はさらに、

対象とする全ての種類の仮想マシン命令に対応する実マシン命令列とこの仮想マシンへの割込み要求に対処するための実マシン命令列を記憶する実マシン命令列記憶部と、

指定された実マシン命令列を実行する実行部とを備え、

前記ブロック判定部は、仮想マシン命令が読み出される度に、それまでに読み出された仮想マシン命令の総数が一定値の倍数に一致するか否か比較し、一致する場合に、前記仮想マシン命令は各ブロックを代表する仮想マシン命令に相当すると判定し、

前記割込み処理部は、前記分岐命令判定部によって各ブロックを代表する仮想マシン命令であると判定される度に、前記実行部に、前記割込み要求に対処するための実マシン命令列を実行させた後に前記仮想マシン命令に対応する実マシン命令列を実行させることを特徴とする請求項１０記載の仮想マシン。

【請求項１２】

実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、

実マシン命令からなる複数のサブプログラムを記憶する実マシンプログラム記

憶手段と、

実行対象となる仮想マシン命令列を記憶する第１領域と、前記実マシンプログラム記憶手段における各サブプログラムへのポインタを記憶する第２領域とを含む命令記憶手段と、

前記命令記憶手段の第１領域から次に実行すべき仮想マシン命令を読み出す読み出し手段と、

読み出された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、

前記解読実行手段は、

読み出された仮想マシン命令が前記第２領域に実行制御を移す命令であるか否かを判定する領域判定部と、

前記第２領域に実行制御を移す命令であると判定された場合に、その移動先の第２領域の箇所に格納された前記ポインタが示す前記サブプログラムを実行するアドレス変換実行部と

を有することを特徴とする仮想マシン。

【請求項１３】

前記命令記憶手段における第１領域と第２領域とは、一定のアドレス値を境界とする隣接した記憶位置に設けられ、

前記領域判定部は、読み出された仮想マシン命令がサブプログラムを呼び出す命令である場合に、その呼び出し先アドレスと前記境界との大小関係を比較することによって前記判定を行うことを特徴とする請求項１２記載の仮想マシン。

【請求項１４】

実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、

実行対象となる仮想マシン命令列を記憶する命令記憶手段と、

前記命令記憶手段から次に実行すべき仮想マシン命令を読み出す読み出し手段と、

読み出された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、

前記命令記憶手段は、前記仮想マシン命令列を構成する基本ブロックそれぞれ

に対応する命令ブロックの集まりからなり、

各命令ブロックは、その命令ブロックの前記命令記憶手段における先頭位置を特定するための識別子を格納した識別子領域と、対応する基本ブロックの非分岐命令だけを格納した非分岐命令領域と、対応する基本ブロックの分岐命令を格納した分岐命令領域とを含み、

前記分岐命令領域に格納された分岐命令は、その分岐先が前記識別子によって指定され、

前記解読実行手段は、読み出された仮想マシン命令が前記分岐命令である場合には、その分岐先として指定されている識別子に対応する命令ブロックの非分岐命令領域の先頭に実行制御を分岐させることを特徴とする仮想マシン。

【請求項１５】

前記解読実行手段は、

次に読み出すべき仮想マシン命令が属する命令ブロックの識別子を格納する識別子レジスタと、その仮想マシン命令のその命令ブロックにおける相対的な記憶位置を示すオフセットを格納するオフセットカウンタとからなるプログラムカウンタを備え、

前記読み出し手段は、前記プログラムカウンタに格納された識別子とオフセットに基づいて前記仮想マシン命令を読み出し、

前記解読実行手段は、読み出された仮想マシン命令が前記分岐命令である場合には、その分岐先として指定されている識別子を前記識別子レジスタに書き込むと共に前記オフセットカウンタを初期値にリセットし、読み出された仮想マシン命令が前記非分岐命令である場合には、前記オフセットカウンタを増加させ、

前記読み出し手段は、前記解読実行手段によって更新されたプログラムカウンタの識別子とオフセットに基づいて次に実行すべき仮想マシン命令を読み出すことを特徴とする請求項１４記載の仮想マシン。

【請求項１６】

前記解読実行手段はさらに、対象とする全ての種類の仮想マシン命令それぞれに対応する実マシン命令列を記憶する実マシン命令列記憶部を備え、

前記命令記憶手段の各命令ブロックはさらに、その命令ブロックの非分岐命令

領域及び分岐命令領域に格納された仮想マシン命令それぞれに対応する実マシン命令列を前記実マシン命令列記憶部に記憶された実マシン命令列から特定するためのデコードデータを含むデコードデータ列を格納したデコードデータ列領域を含み、

前記読み出し手段は、デコードデータ列が格納された命令ブロックについては前記仮想マシン命令に代えて前記デコードデータを読み出し、デコードデータ列が格納されていない命令ブロックについては仮想マシン命令を読み出した後に、その仮想マシン命令に対応する実マシン命令列を前記実マシン命令列記憶部に記憶された実マシン命令列から特定するためのデコードデータを生成し、

前記解読実行手段は、前記読み出し手段から読み出し又は生成されたデコードデータによって特定される実マシン命令列を前記実マシン命令列記憶部から読み出して実行することを特徴とする請求項１５記載の仮想マシン。

【請求項１７】

前記命令記憶手段の各命令ブロックはさらに、その命令ブロックのデコードデータ列領域に前記デコードデータが格納されているか否かを示すフラグを格納したフラグ領域を含み、

前記解読実行手段はさらに、分岐命令を実行した場合には、その分岐先となる命令ブロックのフラグ領域に格納されたフラグを読み出して保持するカレントフラグ記憶部を備え、

前記読み出し手段は、前記カレントフラグ記憶部に保持されたフラグに基づいて、デコードデータの読み出し又は仮想マシン命令の読み出しを行うことを特徴とする請求項１６記載の仮想マシン。

【請求項１８】

前記命令記憶手段の各命令ブロックはさらに、その命令ブロックのデコードデータ列領域に前記デコードデータが格納されているか否かを示すフラグを格納したフラグ領域を含み、

前記解読実行手段はさらに、分岐命令を実行した場合にその分岐先となる命令ブロックのフラグ領域に格納されたフラグを参照することにより、その命令ブロックにはデコードデータが格納されていないと判定したときには、その命令ブロ

ックに格納されている仮想マシン命令列を読み出して対応するデコードデータ列に変換し、その命令ブロックのデコードデータ列領域に書き込むデコードデータ列書込み部を備えることを特徴とする請求項１６記載の仮想マシン。

【請求項１９】

実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、

実行対象となる仮想マシン命令列を圧縮符号で記憶する命令記憶手段と、

前記命令記憶手段から次に実行すべき仮想マシン命令の圧縮符号を読み出し、対応する仮想マシン命令に復号する読み出し手段と、

復号された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、

前記命令記憶手段は、前記仮想マシン命令列を構成する基本ブロックそれぞれに対応する命令ブロックの集まりからなり、

各命令ブロックは、その命令ブロックの前記命令記憶手段における先頭位置を特定するための識別子を格納した識別子領域と、対応する基本ブロックの非分岐命令だけを格納した非分岐命令領域と、対応する基本ブロックの分岐命令を格納した分岐命令領域とを含み、

前記分岐命令領域に格納された分岐命令は、その分岐先が前記識別子によって指定され、

前記解読実行手段は、復号された仮想マシン命令が前記分岐命令である場合には、その分岐先として指定されている識別子に対応する命令ブロックの非分岐命令領域の先頭に実行制御を分岐させることを特徴とする仮想マシン。

【請求項２０】

前記命令記憶手段の各命令ブロックにはさらに、その命令ブロックに格納された仮想マシン命令の圧縮符号を伸長復号するための情報テーブルであって、対応する圧縮符号と仮想マシン命令との組の集まりからなる復号テーブルを格納した復号テーブル領域を含み、

前記読み出し手段は、次に実行すべき仮想マシン命令が属する命令ブロックの復号テーブルを参照しながら、前記命令記憶手段から読み出した圧縮符号を対応する仮想マシン命令に復号することを特徴とする請求項１９記載の仮想マシン。

【請求項２１】

前記解読実行手段は、

次に読み出すべき仮想マシン命令が属する命令ブロックの識別子を格納する識別子レジスタと、その仮想マシン命令のその命令ブロックにおける相対的な記憶位置を示すオフセットを格納するオフセットカウンタとからなるプログラムカウンタを備え、

前記読み出し手段は、前記プログラムカウンタに格納された識別子とオフセットに基づいて前記仮想マシン命令を読み出し、

前記解読実行手段は、読み出された仮想マシン命令が前記分岐命令である場合には、その分岐先として指定されている識別子を前記識別子レジスタに書き込むと共に前記オフセットカウンタを初期値にリセットし、読み出された仮想マシン命令が前記非分岐命令である場合には、前記オフセットカウンタを増加させ、

前記読み出し手段は、前記解読実行手段によって更新されたプログラムカウンタの識別子とオフセットに基づいて次に実行すべき仮想マシン命令を読み出すことを特徴とする請求項２０記載の仮想マシン。

【請求項２２】

前記解読実行手段はさらに、対象とする全ての種類の仮想マシン命令それぞれに対応する実マシン命令列を記憶する実マシン命令列記憶部を備え、

前記命令記憶手段の各命令ブロックはさらに、その命令ブロックの非分岐命令領域及び分岐命令領域に格納された仮想マシン命令それぞれに対応する実マシン命令列を前記実マシン命令列記憶部に記憶された実マシン命令列から特定するためのデコードデータを含むデコードデータ列を格納したデコードデータ列領域を含み、

前記読み出し手段は、デコードデータ列が格納された命令ブロックについては前記仮想マシン命令に代えて前記デコードデータを読み出し、デコードデータ列が格納されていない命令ブロックについては仮想マシン命令の圧縮符号を読み出して対応する仮想マシン命令に復号した後に、その仮想マシン命令に対応する実マシン命令列を前記実マシン命令列記憶部に記憶された実マシン命令列から特定するためのデコードデータを生成し、

前記解読実行手段は、前記読み出し手段から読み出し又は生成されたデコードデータによって特定される実マシン命令列を前記実マシン命令列記憶部から読み出して実行することを特徴とする請求項２１記載の仮想マシン。

【請求項２３】

前記命令記憶手段の各命令ブロックはさらに、その命令ブロックのデコードデータ列領域に前記デコードデータが格納されているか否かを示すフラグを格納したフラグ領域を含み、

前記解読実行手段はさらに、分岐命令を実行した場合には、その分岐先となる命令ブロックのフラグ領域に格納されたフラグを読み出して保持するカレントフラグ記憶部を備え、

前記読み出し手段は、前記カレントフラグ記憶部に保持されたフラグに基づいて、デコードデータの読み出し又は仮想マシン命令の読み出しを行うことを特徴とする請求項２２記載の仮想マシン。

【請求項２４】

前記命令記憶手段の各命令ブロックはさらに、その命令ブロックのデコードデータ列領域に前記デコードデータが格納されているか否かを示すフラグを格納したフラグ領域を含み、

前記解読実行手段はさらに、分岐命令を実行した場合にその分岐先となる命令ブロックのフラグ領域に格納されたフラグを参照することにより、その命令ブロックにはデコードデータが格納されていないと判定したときには、その命令ブロックに格納されている仮想マシン命令列の圧縮符号を読み出して対応する仮想マシン命令列に復号した後に、対応するデコードデータ列に変換し、その命令ブロックのデコードデータ列領域に書き込むデコードデータ列書込み部を備えることを特徴とする請求項２２記載の仮想マシン。

【請求項２５】

実マシンによる制御の下で仮想マシン命令を実行する仮想マシンと共に用いられ、実行対象となる一部の仮想マシン命令列をその実行に先立って実マシン命令列に変換するその場式コンパイラであって、

前記仮想マシン命令列を構成する仮想マシン命令それぞれについて、前記仮想

マシン命令列を基本ブロックに分割した場合の各基本ブロックの先頭となる仮想マシン命令であるか否かを示すブロック先頭情報の入力を受け付けるブロック先頭情報獲得手段と、

前記仮想マシン命令列を構成する仮想マシン命令それぞれについて、対応する実マシン命令列に変換する変換手段と、

前記変換手段によって得られた実マシン命令列における実マシン命令であって、前記ブロック先頭情報獲得手段が受け付けたブロック先頭情報によって特定された基本ブロックの先頭となる仮想マシン命令に対応する実マシン命令が、前記実マシンによってアドレシングされ得ない位置に配置される命令であるか否かを判定する分岐違反判定手段と、

前記分岐違反判定手段によって前記実マシン命令が前記位置に配置される命令であると判定された場合に、前記実マシン命令が前記位置に配置されないよう１個以上の無動作命令を前記実マシン命令列に追加挿入して出力する出力手段と

を備えることを特徴とするコンパイラ。

【請求項２６】

前記出力手段は、前記実マシン命令が前記実マシンによってアドレシングされ得る後続位置にずらせるために必要な個数の無動作命令を前記実マシン命令列における基本ブロックの先頭位置に追加挿入することを特徴とする請求項２５記載のコンパイラ。

【請求項２７】

実マシンによる制御の下で仮想マシン命令を実行するスタック型仮想マシンにおいて、実行対象となる仮想マシン命令列を格納しておくための命令記憶手段における前記格納方法であって、

実行対象となる仮想マシン命令列と、それら仮想マシン命令それぞれに対応づけられた情報であって、対応づけられた仮想マシン命令に後続する仮想マシン命令が実行された場合の前記スタックでのデータの格納状態の変化を示す先行命令情報とを関連付けて格納することを特徴とする仮想マシン命令列の格納方法。

【請求項２８】

コンピュータをスタック型仮想マシンとして機能させるためのプログラムを記

録した記録媒体であって、

前記仮想マシンは、

データを後入れ先出し方式で一時的に記憶するためのスタック手段と、

実行対象となる仮想マシン命令列と、それら仮想マシン命令それぞれに対応づけられた情報であって、対応づけられた仮想マシン命令に後続する仮想マシン命令が実行された場合の前記スタック手段でのデータの格納状態の変化を示す先行命令情報とを記憶する命令記憶手段と、

前記命令記憶手段から次に実行すべき仮想マシン命令及び対応する先行命令情報を読み出す読み出し手段と、

読み出された仮想マシン命令と先行命令情報との組み合わせに対応する演算処理を特定し実行する解読実行手段と

を備えることを特徴とする記録媒体。

【請求項２９】

実マシンによる制御の下で仮想マシン命令を実行する仮想マシンにおいて、実行対象となる仮想マシン命令列を格納しておくための命令記憶手段における前記格納方法であって、

前記命令記憶手段は、前記仮想マシン命令列を構成する基本ブロックそれぞれに対応する命令ブロックの集まりからなり、

各命令ブロックは、その命令ブロックの前記命令記憶手段における先頭位置を特定するための識別子を格納した識別子領域と、対応する基本ブロックの非分岐命令だけを格納した非分岐命令領域と、対応する基本ブロックの分岐命令を格納した分岐命令領域とを含み、

前記分岐命令領域に格納された分岐命令は、その分岐先が前記識別子によって指定されていることを特徴とする仮想マシン命令列の格納方法。

【請求項３０】

コンピュータを仮想マシンとして機能させるためのプログラムを記録した記録媒体であって、

前記仮想マシンは、

実行対象となる仮想マシン命令列を記憶する命令記憶手段と、

前記命令記憶手段から次に実行すべき仮想マシン命令を読み出す読み出し手段と、

読み出された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、

前記命令記憶手段は、前記仮想マシン命令列を構成する基本ブロックそれぞれに対応する命令ブロックの集まりからなり、

各命令ブロックは、その命令ブロックの前記命令記憶手段における先頭位置を特定するための識別子を格納した識別子領域と、対応する基本ブロックの非分岐命令だけを格納した非分岐命令領域と、対応する基本ブロックの分岐命令を格納した分岐命令領域とを含み、

前記分岐命令領域に格納された分岐命令は、その分岐先が前記識別子によって指定され、

前記解読実行手段は、読み出された仮想マシン命令が前記分岐命令である場合には、その分岐先として指定されている識別子に対応する命令ブロックの非分岐命令領域の先頭に実行制御を分岐させることを特徴とする記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、仮想マシン及び仮想マシンコンパイラに関し、特に仮想マシンの実行速度を向上させる技術に関する。

【０００２】

【従来の技術】

（仮想マシンの一般技術）

ＣＰＵ（以下、「実マシン」と呼ぶ。）の種別に依存することなく、同一のプログラムをそのまま、各種ＣＰＵを備えるパーソナルコンピュータやワークステーション等のコンピュータ上で実行させる技術として仮想マシンがある。情報通信分野、特に、異機種のコンピュータ同士が接続されるネットワークにおいては、共有資源の迅速かつ円滑な利用のために、このような実マシンのアーキテクチャの差異を吸収する技術である仮想マシンの果たす役割は大きい。

【０００３】

仮想マシンは、仮想マシン固有の命令（以下、「仮想マシン命令」と呼ぶ。）の集まりからなる実行形式のプログラム（以下、「仮想マシンプログラム」又は「仮想マシン命令列」と呼ぶ。）をソフトウェア的に解読・実行する仮想的なプロセッサであり、一般に、仮想マシン自体は、実マシン用の命令（以下、「実マシン命令」と呼ぶ。）で書かれたプログラム（以下、「実マシンプログラム」又は「実マシン命令列」と呼ぶ。）で実現されている。そして、仮想マシンの中心的な課題である実行速度の確保という点から、仮想マシン自体のアーキテクチャとしてはスタック型（スタックマシン）が採られる。

【０００４】

従来のスタック型仮想マシンとして、米国 SunMicrosystems,Inc.社が開発したJava（登録商標）がある。

図７１は、Javaのような従来のスタック型仮想マシン４４００の構成を示す機能ブロック図である。従来の仮想マシン４４００は、実行対象となる仮想マシンプログラムを保持する命令格納部４４０１、その命令格納部４４０１から仮想マシン命令を読み出して解読するデコード部４４０２、その解読結果であるデコードデータに従ってオペレーションを実行する実行部４４１０及びその実行のための一時的な後入れ先出し方式の記憶領域であるスタック４４２０から構成される。なお、本図において、実線はデータの流れ、点線は制御の流れを示す。

【０００５】

デコード部４４０２は、さらに、この仮想マシン４４００が実行する全ての仮想マシン命令それぞれに対応するマイクロプログラム（これは、実行部４４１０に格納されている。）へのジャンプアドレス等を予め記憶するデコードテーブル４４０６、命令格納部４４０１から次に読み出すべき命令のアドレスを保持するプログラムカウンタ（ＰＣ）４４０４、その命令を読み出す読込み部４４０３及び読み出した命令に対してデコードテーブル４４０６を参照することにより対応するジャンプアドレスを特定して実行部４４１０に出力する検索部４４０５を有する。ここで、マイクロプログラムとは、個々の仮想マシン命令のオペレーションに対応する実マシンプログラムを意味する。

【０００６】

実行部４４１０は、上記マイクロプログラム、つまり、全ての仮想マシン命令それぞれに対応する実マシンプログラムを予め上記ジャンプアドレスで始まる箇所に記憶しているマイクロプログラム記憶部４４１１及びスタック４４２０の最上位アドレスを示すスタックポインタ（ＳＰ）４４１２を有する。

図７２は、この仮想マシン４４００の命令セットを説明するための図である。本図には、この仮想マシン４４００が解読・実行する全ての仮想マシン命令のニーモニック表示、オペレーションの意味、スタックの状態変化及び実行後のスタックポインタの変化が示されている。本図において、表記"s0"はスタックの最上位の値を表し、"s1"はスタックの最上位から２番目の値を表す。例えば仮想マシン命令"Add"の表記"s0←s0+s1"は、実行後にスタックの最上位の値が、実行前のスタックの最上位の値と、スタックの最上位から２番目の値を加えたものとなることを意味する。表記"sp←sp-1"は、実行後にスタックの階層が一つ減ることを意味する。

【０００７】

図７３は、図７１に示されたデコードテーブル４４０６の内容を示す図である。各仮想マシン命令の種別を示すオペコード４４０６ａと、その仮想マシン命令に対応するマイクロプログラム記憶部４４１１内のマイクロプログラムへのジャンプアドレス４４０６ｂと、その仮想マシン命令に含まれるオペランド数４４０６ｃとが格納されている。なお、全てのオペコードは１バイト長とし、オペランドの単位も１バイトとする。また、物理的なビット配列で表現された仮想マシン命令（オペコードだけやオペランドだけを指すこともある）を「仮想マシンコード」と呼ぶ。

【０００８】

図７４（ａ）～（ｄ）は、それぞれ、図７１に示されたマイクロプログラム記憶部４４１１に格納されているマイクロプログラムのうち、仮想マシン命令"Push"、"Add"、"Mult"に対応するもの及びそれらに共通する後半部分（次の仮想マシン命令にジャンプするための実マシンプログラム）のリストを示す。ここで用いられている実マシン命令の意味は、図７５に示される通りである。つまり、こ

の仮想マシン４４００自体は、図７５に示される実マシン命令を解読・実行する実マシンによって実行されるものとしている。なお、物理的には、仮想マシン４４００のＰＣ４４０４は実マシンのレジスタ２番（ｒ２）に、仮想マシン４４００のＳＰ４４１２は実マシンのレジスタ３番（ｒ３）に割り当てられている。

【０００９】

図７６は、図７１に示されたデコード部４４０２の動作を示すフローチャートである。読込み部４４０３は、実行部４４１０から信号線Ｒを介して「次の命令を読み込む」旨の指示を受けると（ステップ４５０２～４５０３）、ＰＣ４４０４に格納されたアドレスに基づいて命令格納部４４０１に格納された仮想マシン命令を読み出す（ステップ４５０４～４５０５）。続いて、検索部４４０５は、デコードテーブル４４０６を参照することにより、その仮想マシン命令に対応するジャンプアドレスと必要なオペランドとをデコードデータとして実行部４４１０に出力すると共に（ステップ４５０６）、１個の仮想マシン命令に対する解読の区切りを示す旨の通知「読込み終了」を信号線Ｒを介して実行部４４１０に通知する（ステップ４５０７）。

【００１０】

図７７は、図７６のフローチャートにおけるステップ４５０６の詳細を示す図である。検索部４４０５は、まず、読込み部４４０３によって読み込まれた１バイト分の仮想マシンコード（オペコード）とデコードテーブル４４０６中のオペコード４４０６ａとを逐一に比較し（ステップ４８０２～４８０７）、一致した場合には、対応するジャンプアドレス４４０６ｂとオペランド数４４０６ｃとをデコードテーブル４４０６から読み出す。そして、そのジャンプアドレス４４０６ｂを実行部４４１０に出力した後に（ステップ４８０８）、そのオペランド数４４０６ｃに相当するオペランドを読込み部４４０３を介して命令格納部４４０１から読み出し、実行部４４１０に出力する（ステップ４８０９～４８１３）。

【００１１】

なお、図７６及び図７７に示されたフローチャートは、デコード部４４０２から出力されたデコードデータが直接実行部４４１０に渡されることを前提とするものであるが、複数のデコードデータを記憶できるバッファを介して実行部４４

１０に渡されることを前提とした場合には、デコード部４４０２の動作は、図７８に示されるフローチャートの通りとなる。つまり、バッファの記憶容量が許す限り、実行部４４１０による実行とは独立して、命令格納部４４０１からの仮想マシン命令の読み出しと解読とを行う（ステップ４６０５～４６１３）。

【００１２】

図７９は、図７１に示された実行部４４１０の動作を示すフローチャートである。実行部４４１０は、ＳＰ４４１２とＰＣ４４０４とを初期化しておいた後に（ステップ４７０２）、個々の仮想マシン命令ごとに以下の動作を繰り返す（ステップ４７０３～４７０７）。つまり、読込み部４４０３に対して信号線Ｒを介して「次の命令を読み込む」旨の指示を通知した後に（ステップ４７０３）、その信号線Ｒから「読込み終了」の通知を受けるまで、検索部４４０５から送られてくるデコードデータを読込み、そのデコードデータに含まれるジャンプアドレスにジャンプすることで、その仮想マシン命令に相当するマイクロプログラム記憶部４４１１内のマイクロプログラムを実行する（ステップ４７０４～４７０７）。

【００１３】

図８０（ａ）は、従来の仮想マシン４４００の具体的な動作例を説明するためのサンプルプログラムのリストを示す。つまり、命令格納部４４０１には、図８０（ｂ）に示される演算式「2*(3+4)」を計算する仮想マシンプログラムが置かれているとする。

図８０（ｃ）は、図８０（ａ）に示された仮想マシンプログラムが従来の仮想マシン４４００によって解読・実行された場合において、デコード部４４０２から順次出力されるデコードデータを示す図である。つまり、デコード部４４０２は、解読した仮想マシン命令に相当するマイクロプログラムのジャンプアドレスと必要なオペランドをデコードデータとして順次実行部４４１０に渡す。

【００１４】

図８１（ａ）及び（ｂ）は、それぞれ、図８０（ｃ）に示されたデコードデータ列に従って実行部４４１０が図８０（ａ）の仮想マシンプログラムを実行した場合における各仮想マシン命令が実行される前後でのＰＣ４４０４、ＳＰ４４１

２及びスタック４４２０の一連の変化の前半部と後半部を示す図である。ここで、ＰＣ４４０４は、図８０（ａ）に示された仮想マシンプログラム中の次に実行すべき仮想マシン命令の左欄に付されたアドレスを差し、その初期値は「１」である。また、ＳＰ４４１２は、実行部４４１０が最後に格納又は読み出したスタック４４２０の最上位の位置を示し、その初期値はスタック４４２０が空であることを示す「－１」である。図８１（ａ）及び（ｂ）から分かるように、ＰＣ４４０４が９に達した段階で、計算式"2*(3+4)"の値が算出されている。

【００１５】

以上のような仮想マシン４４００の中心的な課題は実行速度の向上である。つまり、仮想マシンは、仮想マシン命令を解読する等の処理のためにオーバーヘッドを生じるので、実マシンによる実マシンプログラムの直接的な実行の場合に比較し、大幅に実行速度が劣る。そこで、仮想マシンの実行速度を向上させる従来の技術として、以下の手法が提案されている。

（第１の従来技術）

第１の従来技術は、スタックの最上位の記憶域（以下、「Top Of Stack（TOS）変数」と呼ぶ。）をメモリ上ではなく、実マシンの特定のレジスタに割り当てておく手法である（「ＰＬＤＩ」ＡＣＭ 1995,pp315-327　参照）。

【００１６】

図８２（ａ）～（ｄ）は、この従来技術に基づいてマイクロプログラム記憶部を実装した場合の主な仮想マシン命令についてのマイクロプログラムのリストであり、上述の仮想マシン４４００における図７４に対応する。ここでは、物理的には、TOS変数は、実マシンのレジスタ０番（ｒ０）に割り当てられている。また、ＰＣ４４０４及びＳＰ４４１２は、図７４（ａ）～（ｄ）と同様に、それぞれレジスタ２番（ｒ２）及びレジスタ３番（ｒ３）に割り当てられている。

【００１７】

図８３（ａ）及び（ｂ）は、図８２（ａ）～（ｄ）に示されたマイクロプログラムを備える仮想マシンによって図８０（ａ）の仮想マシンプログラムが実行された場合のＰＣ４４０４、ＳＰ４４１２、TOS変数４４２１及びスタック４４２０のうちメモリに割り当てられた領域４４２２（以下、仮想マシンのスタックの

うち、メモリに割り当てられた領域を「メモリスタック」と呼ぶ。）の一連の変化の前半部と後半部を示す図であり、上述の仮想マシン４４００における図８１（ａ）及び（ｂ）に対応する。この手法によっても、ＰＣが９に達した段階で、計算式"2*(3+4)"の値が算出されていることが分かる。

【００１８】

図７４（ａ）～（ｄ）と図８２（ａ）～（ｄ）とを比較して分かるように、この第１の従来技術によれば、メモリへのアクセスの回数が減少される。加算"Add"や乗算"Mult"等の仮想マシン命令の実行においては、上述の仮想マシン４４００によれば、各演算のためにスタック４４２０からの２回の読み出しと１回の書き込みの合計３回のメモリアクセスが必要とされたが、この従来技術によれば、TOS変数がメモリ上ではなくレジスタに割り当てられているので、必要なメモリアクセスはメモリスタック４４２２からの１回の読み出しだけで済む。従って、メモリアクセスが減少された分だけ実行速度が向上され得る。

（第２の従来技術）

第２の従来技術は、「ネイティブコード化」と呼ばれている方法であり、仮想マシンプログラムの特定箇所を実マシン命令で記述しておき、その特定箇所については実マシンによって直接に実行させるという手法である。そのために、その特定箇所が実マシン命令で記述されている旨の識別子が用いられる。

【００１９】

例えば、上述のJavaでは、仮想マシンプログラムを収めるクラスファイルのアクセスフラグ(構造体"method_info"の要素である16ビットのフラグ"access_flags"等)に定数名"ACC_NATIVE"(256)を格納しておくことで、そのプログラム部分は実マシン命令で実装されていることを示す（米国 SunMicrosystems,Inc.社発行のJavaバイトコード及びJava仮想マシンの仕様書(1995年版)参照）。

【００２０】

この従来技術によれば、特定箇所のプログラムについては、実マシンによって直接に実行されるので、その分だけ実行速度は向上され得る。

（第３の従来技術）

第３の従来技術は、「その場式コンパイラ(Just-In-Time Compiler)」と呼ば

れる手法で、仮想マシンプログラムの実行時において、必要に応じてその一部をコンパイル、即ち、実マシン命令に置き換える手法である（ローラ・リメイほか「Java言語入門」（株）プレンティスホール出版，1996年　参照）。この従来技術によれば、既にコンパイルが完了しているプログラム部分については、実マシンにより直接に実行されるので、その分だけ実行速度が向上され得る。

（第４の従来技術）

第４の従来技術は、ネットワーク上のサーバコンピュータに存在する仮想マシンプログラムを自らのコンピュータにダウンロードした後に実行するような形態において、予め仮想マシンプログラムのコードをLZ法やハフマン符号化などによって圧縮しておくことで、転送時間を削減する手法である（特開平７－１２１３５２や特開平８－２６３２６３など）。

【００２１】

この従来技術によれば、仮想マシンプログラムの転送時間の全体の処理時間に占める割合が大きい場合には、全体としての実行速度が向上され得る。

【００２２】

【発明が解決しようとする課題】

しかしながら、上記従来技術には以下の問題点がある。

（第１の従来技術の問題点）

上記第１の従来技術、つまり、TOS変数を実マシンのレジスタに割り当てておく手法には、近年安価になってきたスーパースカラアーキテクチャの実マシン（以下、「スーパースカラマシン」と呼ぶ。）には向いていないという問題点、つまり、並列処理機能を有しない通常の実マシン（以下、「通常マシン」と呼ぶ。）で実行する場合と比較し、大きくは実行速度が向上されないという問題点がある。以下、その問題点を説明する。

【００２３】

まず、図８４～図９２を用いて、上記スーパースカラマシンや通常マシンなどのレジスタマシンが有するパイプラインの一般的な動作及びその表記法を説明する。

図８４は、パイプラインの各ステージに対する省略記号を説明する図である。

ここでは、スーパースカラマシン及び通常マシンは、いずれも５個のステージからなるパイプラインを有するとする。

【００２４】

図８５は、通常マシンのパイプラインの理想的な流れを示す図である。ここでは、４つの実マシン命令が１クロックごとに隙間なく順次に処理される様子が示されている。この状態では、パイプラインの各ステージは並列に動作するので、平均して、１クロックで１つの実マシン命令が実行されることになる。

一方、図８６は、スーパースカラマシンのパイプラインの理想的な流れを示す図である。ここでは、スーパースカラマシンは、２本の独立したパイプラインを有するものとしている。この状態では、平均して、１クロックで２つの実マシン命令が実行されることになる。つまり、通常マシンに比べ、２倍の処理能力を持つ。

【００２５】

図８７は、通常マシンにおいて、パイプラインにハザード（干渉）が生じた場合のパイプラインの流れを示す図である。ここでは、命令Ｂは、先行する命令Ａに対して真のデータ依存関係（true dependency又はdata dependensyと呼ばれる関係）を持つ、つまり、命令Ａによる演算結果を用いる命令としている。すると、命令Ａによる演算結果は、メモリ参照ステージMEMが完了しないと明らかにならないため、直後の命令Ｂは、命令ＡのステージMEMの完了まで実行が一時的に保留される結果、ここにハザード（図中の記号"-"で示されるステージ）が生じる。なお、パイプライン構造を採る実マシンでは、先行する命令の実行が保留されると、後続の命令も次のステージに移ることができないため、本図に示されるように、命令Ｂの直後の命令Ｃも実行が一時的に保留される。

【００２６】

一方、図８８は、スーパースカラマシンにおいて、パイプラインにハザードが生じた場合のパイプラインの流れを示す図である。ここでは、命令Ｂ１は、先行する命令Ａ１及び命令Ａ２に対して真のデータ依存関係を持つとしている。なお、本図において、命令Ｃ２の５クロック目にハザードが生じているのは、先行する命令Ｂ１と命令Ｃ１とによって実行に必要な２個のユニット（ALU）が使用さ

れているからである。

【００２７】

図８９及び図９０は、それぞれ、図８７及び図８８の場合において、メモリ参照が行われてからその値が次に使用されるまでに２クロックの時間を必要とするときのパイプラインの流れを示す図である。実際の実マシンでは、一次キャッシュのメモリから、値を得るまでに２クロックの時間を要する場合が多いからである。なお、二次キャッシュ等にアクセスする場合は、より多くのクロック数を要する。

【００２８】

図９１及び図９２は、それぞれ、通常マシン及びスーパースカラマシンにおいて、命令Ａ及び命令Ａ２がレジスタでジャンプ先を指定する命令である場合のパイプラインの流れを示す図である。ジャンプ先は、それぞれ、命令Ａ及び命令Ａ２のレジスタ参照ステージRFが終了しないと判明しないので、その間に無駄にフェッチされた後続命令Ｂは、次の第３クロック（記号"x"で示されるステージ）おいてキャンセルされる。従って、ジャンプ先の命令Ｃ及び命令Ｃ１は、それぞれ命令Ａ及び命令Ａ２のレジスタ参照ステージRFが終了した後で実行される。

【００２９】

次に、図９３～図９６を用いて、上記第１の従来技術に係るスーパースカラマシン及び通常マシンの具体的な問題点を説明する。

図９３～図９６は、いずれも、第１の従来技術に係る仮想マシンが図８０（ａ）に示された仮想マシンプログラムを実行した場合の実マシンのパイプラインの流れ、より具体的には、アドレス７の仮想マシン命令"Add"のマイクロプログラム（図８２（ａ））の後半部分（図８２（ｄ）に示されたジャンプ処理）と、続くアドレス８の仮想マシン命令"Mult"のマイクロプログラム（図８２（ｃ））の前半部分（乗算計算処理）におけるパイプラインの流れを示す。但し、図９３～図９６は、それぞれ、メモリ参照が行われてからその値が次に使用されるまでに１クロックの時間で済む場合における通常マシン、その場合におけるスーパースカラマシン、メモリ参照が行われてからその値が次に使用されるまでに２クロックの時間を必要とする場合における通常マシン、その場合におけるスーパースカ

ラマシンのパイプラインの流れを示す。

【００３０】

これら一連のマイクロプログラム（図８２（ｄ）及び図８２（ａ））には、重大な真のデータ依存関係が二つ存在する。つまり、仮想マシン命令"Add"のジャンプ処理（図８２（ｄ））におけるジャンプアドレスの読込み命令"Load"とジャンプ命令"Jmp"との間に存在する第１の依存関係と、仮想マシン命令"Mult"の計算処理（図８２（ｃ））におけるスタックの２番目の変数を読み込む命令"Load"と乗算命令"Mult"との間に存在する第２の依存関係である。

【００３１】

図９３に示されたパイプラインでは、上記第１の依存関係は、実マシン命令"Load"と実マシン命令"Jump"間に挿入された実マシン命令"Inc"で吸収され、さらに上記第２の依存関係も、実マシン命令"Load"と実マシン命令"Mult"間に挿入された実マシン命令"Dec"で吸収されるため、一連の処理において生じるパイプラインの乱れは、実マシン命令"Jmp"の実行に起因する１個の命令のキャンセルだけとなり、処理全体は１１クロックで完了する。

【００３２】

図９４に示されたパイプラインでは、上記第１及び第２の依存関係は吸収されないために、一連の処理において生じるパイプラインの乱れは、上記第１の依存関係に起因する第４クロックのハザードと、実マシン命令"Jmp"の実行に起因する５個の命令のキャンセルと、上記第２の依存関係に起因する第８クロックのハザードの合計３個所となり、処理全体は、図９３の場合と同様に、１１クロックを要することになる。

【００３３】

図９５に示されたパイプラインでは、図９４の場合と同様に上記第１及び第２の依存関係は吸収ず、一連の処理において生じるパイプラインの乱れは、上記第１の依存関係に起因する第５クロックのハザードと、実マシン命令"Jmp"の実行に起因する１個の命令のキャンセルと、上記第２の依存関係に起因する第１０クロックのハザードの合計３個所となり、処理全体は１３クロックで完了する。

【００３４】

図９６に示されたパイプラインでは、図９４の場合と同様に上記第１及び第２の依存関係は吸収ず、一連の処理において生じるパイプラインの乱れは、上記第１の依存関係に起因する第４及び第５クロックのハザードと、実マシン命令"Jmp"の実行に起因する７個の命令のキャンセルと、上記第２の依存関係に起因する第９及び第１０クロックのハザードの合計３個所となり、処理全体は、図９５の場合と同様に、１３クロックを要することになる。

【００３５】

上記図９３及び図９４に示された一連の処理はいずれも１１クロックを要していること、及び、上記図９５及び図９６に示された一連の処理はいずれも１３クロックだけ要していることから分かるように、この従来技術によれば、通常マシンによる場合とスーパースカラマシンによる場合とで実行時間に差がなく、スーパースカラマシンによる並列処理の長所が生かされていない。

【００３６】

このように、この従来技術には、スーパースカラマシンによる実行効率が大幅に低下するという問題点があるが、これに加えて、エラーなどの例外処理やデバッグに不可欠な割込み処理の実現方法が提供されていないという問題点もある。

そのために、一般の実マシンと同様に、１個の仮想マシン命令を実行する度に割込み状態の検出と処理を行うことになるが、それでは、仮想マシン命令を実行する度に新たなメモリアクセス（割込み状態を保持するメモリ上の変数をレジスタに読み込むためのデータ転送）が加わることになり、TOS変数をレジスタに割り当てることによってメモリアクセス回数が削減されるという効果も打ち消されてしまい、全体としての実行速度が向上されない。

（第２の従来技術の問題点）

上記第２の従来技術、つまり、ネイティブコード化と呼ばれている方法には、アーキテクチャの異なる実マシンを対象として仮想マシンプログラムを共有化することが困難になるという問題点がある。この従来技術では、仮想マシンプログラムの一部が特定の実マシンを対象とする実マシン命令で記述されるからである。そのために、例えば、ネットワーク上の異なるアーキテクチャの実マシンを備える５種類のコンピュータによって同じプログラムが共有される場合であれば、

それら５種類の実マシンを対象とする実マシンプログラムを準備しておく必要がある。

【００３７】

また、コンピュータのシステム環境は個々に異なるために、たとえ同じアーキテクチャの実マシンを対象とする場合であっても、必ずしも仮想マシン命令よりも実マシン命令のほうが実行速度の点で優るとも言えない。例えば、プログラムのコードサイズが一般的に大きくなってしまうRISC（縮小型命令セットコンピュータ）タイプの実マシンを対象とする場合では、そのメインメモリのサイズが充分でないときには仮想マシン命令を実マシン命令に置き換えることで仮想メモリのページ入れ替えが頻発し、却って実行速度が低下する結果となる。

（第３の従来技術の問題点）

上記第３の従来技術、つまり、その場式コンパイラと呼ばれる手法でには、仮想マシンプログラムをコンパイルするのに要する時間が長くなってしまうという問題点がある。

【００３８】

その第１の理由は、対象とする実マシン固有のジャンプ先についての制限を満足するための処理が必要とされることである。例えば、ジャンプ先のアドレスがメインメモリのワード（基本語長）境界に制限される実マシンを対象とする場合には、仮想マシン命令をただ単に対応する実マシンプログラムに変換しただけでは、その制限を違反してしまう。

【００３９】

図９７は、この第１の理由を説明するためのサンプル仮想マシンプログラムのリストであり、図９８は、そのフローチャートである。

このサンプル仮想マシンプログラムは、０から９までの合計１０個の整数の和を算出するものであり、初期値の設定（ステップ７００２、アドレス０～６）、計算の終了判断（ステップ７００３、アドレス８～１３）、加算と被加算値の設定（ステップ７００４、アドレス１５～２９）及び終了処理（ステップ７００５、アドレス３１）からなる。

【００４０】

図９９は、この従来技術においてコンパイル時に使用される変換テーブルである。つまり、コンパイルの対象となる仮想マシン命令と、それと置き換えられる実マシンプログラム等との対応表である。なお、この変換テーブルには、参考のために、各実マシンプログラムのコードサイズも示されている。

図１００は、図９９に示された変換テーブルを用いて図９７に示された仮想マシンプログラムをコンパイルした場合に得られる実マシンプログラムのコード配置の様子を図である。ここには、元の仮想マシンに対応する各実マシンプログラムの相対的な配置アドレスが示されている。

【００４１】

ここで、もし、対象とする実マシンが２ワード境界でしかジャンプ先を指定できないとするならば、図１００から明らかなように、アドレス１３の仮想マシン命令"Brz"で指定されるジャンプ先であるアドレス３１の仮想マシン命令"Stop"及びアドレス２９の仮想マシン命令"Br"で指定されるジャンプ先であるアドレス１５の仮想マシン命令"Push"は、いずれも、実マシンプログラムにおける奇数アドレスに配置されるため、即ち、２ワードの境界には配置されないため、ジャンプ先の制限に違反する。従って、そのような制限違反を犯さないための処理が必要とされる。

【００４２】

また、上記問題点を生じる第２の理由は、対象とする実マシンによっては分岐に伴う特別な処理が必要とされることである。例えば、米国SPARCInternational,Inc.社のSPARC(登録商標)アーキテクチャのCPUや米国MIPS Technologies,Inc.社のCPUなど、一部のリスクアーキテクチャのCPUでは、分岐命令の直後に配置されるいくつかの命令のみを特別なルールで実行するものがある。具体的には、分岐の成否にかかわらず後続する所定の命令を実行したり（遅延分岐；delayed branch）、分岐が成立した場合にのみ後続する所定の命令を実行したり（打ち消し分岐；cancellingbranch)する。

【００４３】

このような実マシンを対象とする場合には、一定の解析を行った後に命令の配置をスケジューリングするか、NOP等の無動作命令を分岐命令の直後に配置する

などの特別な処理が必要とされる。

（第４の従来技術の問題点）

上記第４の従来技術、つまり、予め仮想マシンプログラムを圧縮符号化しておく方法では、圧縮された仮想マシンプログラムにおける分岐命令の実行に伴って生じる不具合についての解決手段が示されていないという問題点がある。

【００４４】

図１０１（ａ）は、この問題点を説明するための圧縮符号化テーブルの例であり、可変長符号９３００ａと仮想マシン命令９３００ｂとを対応付けている。図１０１（ｂ）は、図１０１（ａ）に示されたテーブルを用いて仮想マシン命令列Ａ（"babc"）を符号化して得られる符号列である。

図１０１（ｂ）に示された符号列は、その先頭から復号されるならば、元の仮想マシン命令列Ａ（"babc"）に復元され得る。ところが、分岐命令などによって図１０１（ｂ）の途中箇所Ｂに実行制御が移った場合において、その箇所Ｂから始まる符号列"0010110"に対して図１０１（ａ）に示されたテーブルを用いて復号した場合には、誤った仮想マシン命令列"aabc"に復号されてしまうという問題がある。

（第１～４の従来技術に共通の問題点）

上記第１～４の従来技術には、キャッシュ処理を効率化するための技術については提案されていないという共通の問題点がある。そのために、キャッシュメモリを備える実マシンやコンピュータの処理能力を生かした高速な仮想マシンの実現が期待されている。

【００４５】

図１０２は、キャッシュメモリを備える仮想マシンにおいて生じうる問題点を説明するための図であり、仮想マシンのプログラムカウンタ６９０１と命令キャッシュ６９０２の構成を示すブロック図である。命令キャッシュ６９０２は、１０番地分の命令列６９０３をキャッシュブロックとしてキャッシュメモリに存在する全てのキャッシュブロックを特定するためのアドレスを記憶するキャッシュテーブル６９０４を備える。

【００４６】

図１０３は、図９７に示された仮想マシンプログラムがキャッシュメモリに置かれた場合における各キャッシュブロックを区切る境界線Ａ，Ｂ，Ｃを示す図である。本図に示された境界線Ｃから分かるように、単純に仮想マシンプログラムを一定長のキャッシュブロックに分割しただけでは、１個の仮想マシン命令"Br 8"のオペコード"Br"とオペランド"8"とが分断されてしまう。そのために、仮想マシンプログラムのキャッシュブロックへの分割において、プログラムカウンタ６９０１の値を変更する全ての仮想マシン命令について、キャッシュブロックを跨ぐような変更をしていないか判定しなければならず、処理が複雑化し、結果的にキャッシュを導入することで、仮想マシン全体の実行速度が低下してしまう。

【００４７】

ここで、仮想マシンプログラム全体をキャッシュメモリに配置する方法や、その場式コンパイラによる仮想マシンプログラムの内容解析の結果を用いてキャッシュメモリに配置する方法が考えられる。しかし、前者の方法では、キャッシュメモリの利用効率が悪く、ネットワーク環境におけるデータ転送に要する時間が大きくなってしまうという問題があり、また、後者の方法では、キャッシュメモリへの読込み時間が長くかかるという問題があり、いずれの方法によっても、仮想マシン全体としての実行効率は大幅に低下してしまう。

（発明の目的）

そこで、本発明はかかる問題点に鑑みてなされたものであり、従来よりも高速に仮想マシンプログラムを実行する仮想マシン、そのための仮想マシンコンパイラ（以下、仮想マシンと仮想マシンコンパイラとを併せて「仮想マシンシステム」と呼ぶ。）及びその場式コンパイラを提供することを目的とする。ここで、仮想マシンコンパイラとは、Ｃ言語等の高級言語で記述されたソースプログラムを仮想マシンプログラムに翻訳するプログラムである。

【００４８】

この目的を達成するためのより具体的な目的は以下の通りである。

第１に、真のデータ依存関係の影響を吸収することで高速実行を可能とした仮想マシンシステムを提供する。

第２に、割込み処理に基づく実行効率の低下を回避することで高速実行を可能

とした仮想マシンシステムを提供する。

【００４９】

第３に、複数の異なるアーキテクチャの実マシンを対象とする場合であっても、そのために全体としての実行速度を低下させることなく個々の実マシンの環境に応じたネイティブコード化が可能な仮想マシンシステム、即ち、実行速度を低下させることなく実マシンのアーキテクチャへの非依存性を高めた仮想マシンシステムを提供する。

【００５０】

第４に、仮想マシンプログラムのキャッシュブロックへの分割に伴う実行効率の低下や、その場式コンパイラを採用した場合における複雑なアドレス解決に伴う実行効率の低下などを回避することで高速実行を可能としたキャッシュ機構に対応した仮想マシンシステムを提供する。

第５に、分岐命令が含まれている場合であっても圧縮された仮想マシンプログラムを正しく伸長できることが保証された高速実行可能な仮想マシンシステムを提供する。

【００５１】

第６に、複雑なアドレス解決を行う必要のない高速動作が可能なその場式コンパイラを提供する。

【００５２】

【課題を解決するための手段】

上記第１の具体的な目的を達成するために、本発明に係る仮想マシンは、実マシンによる制御の下で仮想マシン命令を実行するスタック型仮想マシンであって、データを後入れ先出し方式で一時的に記憶するためのスタック手段と、実行対象となる仮想マシン命令列と、それら仮想マシン命令それぞれに対応づけられた情報であって、対応づけられた仮想マシン命令に後続する仮想マシン命令が実行された場合の前記スタック手段でのデータの格納状態の変化を示す先行命令情報とを記憶する命令記憶手段と、前記命令記憶手段から次に実行すべき仮想マシン命令及び対応する先行命令情報を読み出す読み出し手段と、読み出された仮想マシン命令と先行命令情報との組み合わせに対応する演算処理を特定し実行する解

読実行手段とを備えることを特徴とする。

【００５３】

また、本発明に係るコンパイラは、スタック型仮想マシンを対象とするコンパイラであって、ソースプログラムを前記仮想マシンが実行する仮想マシン命令列に変換する命令列変換手段と、変換によって得られた仮想マシン命令列を構成する仮想マシン命令それぞれについて、後続する仮想マシン命令が前記仮想マシンによって実行された場合の前記スタック手段でのデータの格納状態の変化を示す先行命令情報を生成する先行命令情報生成手段と、生成された先行命令情報と対応する仮想マシン命令とを関連付けて出力する関連付け手段とを備えることを特徴とする。

【００５４】

上記第２の具体的な目的を達成するために、本発明に係る仮想マシンは、実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、実行対象となる仮想マシン命令列を記憶する命令記憶手段と、前記命令記憶手段から次に実行すべき仮想マシン命令を読み出す読み出し手段と、読み出された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、前記解読実行手段は、読み出された仮想マシン命令が実行制御の流れを分岐させる分岐命令であるか否かを判定する分岐命令判定部と、分岐命令であると判定される度に、その分岐命令の実行に加えて、この仮想マシンへの割込み要求の発生の有無の検出と、発生している場合の割込み処理とを実行する割込み処理部とを有することを特徴とする。

【００５５】

また、本発明に係る仮想マシンは、実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、実行対象となる仮想マシン命令列を記憶する命令記憶手段と、前記命令記憶手段から次に実行すべき仮想マシン命令を読み出す読み出し手段と、読み出された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、前記解読実行手段は、読み出された仮想マシン命令が一定個数の仮想マシン命令の集まりをブロックとした場合の各ブロックを代表する仮想マシン命令に相当するか否かを判定するブロック判定部を備え、各ブロ

ックを代表する仮想マシン命令であると判定される度に、その仮想マシン命令の実行に加えて、この仮想マシンへの割込み要求の発生の有無の検出と、発生している場合の割込み処理とを実行する割込み処理部とを有するとすることもできる。

【００５６】

上記第３の具体的な目的を達成するために、本発明に係る仮想マシンは、実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、実マシン命令からなる複数のサブプログラムを記憶する実マシンプログラム記憶手段と、実行対象となる仮想マシン命令列を記憶する第１領域と、前記実マシンプログラム記憶手段における各サブプログラムへのポインタを記憶する第２領域とを含む命令記憶手段と、前記命令記憶手段の第１領域から次に実行すべき仮想マシン命令を読み出す読み出し手段と、読み出された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、前記解読実行手段は、読み出された仮想マシン命令が前記第２領域に実行制御を移す命令であるか否かを判定する領域判定部と、前記第２領域に実行制御を移す命令であると判定された場合に、その移動先の第２領域の箇所に格納された前記ポインタが示す前記サブプログラムを実行するアドレス変換実行部とを有することを特徴とする。

【００５７】

ここで、前記命令記憶手段における第１領域と第２領域とは、一定のアドレス値を境界とする隣接した記憶位置に設けられ、前記領域判定部は、読み出された仮想マシン命令がサブプログラムを呼び出す命令である場合に、その呼び出し先アドレスと前記境界との大小関係を比較することによって前記判定を行うとすることもできる。

【００５８】

上記第４の具体的な目的を達成するために、本発明に係る仮想マシンは、実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、実行対象となる仮想マシン命令列を記憶する命令記憶手段と、前記命令記憶手段から次に実行すべき仮想マシン命令を読み出す読み出し手段と、読み出された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、前記命令記憶

手段は、前記仮想マシン命令列を構成する基本ブロックそれぞれに対応する命令ブロックの集まりからなり、各命令ブロックは、その命令ブロックの前記命令記憶手段における先頭位置を特定するための識別子を格納した識別子領域と、対応する基本ブロックの非分岐命令だけを格納した非分岐命令領域と、対応する基本ブロックの分岐命令を格納した分岐命令領域とを含み、前記分岐命令領域に格納された分岐命令は、その分岐先が前記識別子によって指定され、前記解読実行手段は、読み出された仮想マシン命令が前記分岐命令である場合には、その分岐先として指定されている識別子に対応する命令ブロックの非分岐命令領域の先頭に実行制御を分岐させることを特徴とする。

【００５９】

上記第５の具体的な目的を達成するために、本発明に係る仮想マシンは、実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、実行対象となる仮想マシン命令列を圧縮符号で記憶する命令記憶手段と、前記命令記憶手段から次に実行すべき仮想マシン命令の圧縮符号を読み出し、対応する仮想マシン命令に復号する読み出し手段と、復号された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、前記命令記憶手段は、前記仮想マシン命令列を構成する基本ブロックそれぞれに対応する命令ブロックの集まりからなり、各命令ブロックは、その命令ブロックの前記命令記憶手段における先頭位置を特定するための識別子を格納した識別子領域と、対応する基本ブロックの非分岐命令だけを格納した非分岐命令領域と、対応する基本ブロックの分岐命令を格納した分岐命令領域とを含み、前記分岐命令領域に格納された分岐命令は、その分岐先が前記識別子によって指定され、前記解読実行手段は、復号された仮想マシン命令が前記分岐命令である場合には、その分岐先として指定されている識別子に対応する命令ブロックの非分岐命令領域の先頭に実行制御を分岐させることを特徴とする。

【００６０】

上記第６の具体的な目的を達成するために、本発明に係るその場式コンパイラは、実マシンによる制御の下で仮想マシン命令を実行する仮想マシンと共に用いられ、実行対象となる一部の仮想マシン命令列をその実行に先立って実マシン命

令列に変換するその場式コンパイラであって、前記仮想マシン命令列を構成する仮想マシン命令それぞれについて、前記仮想マシン命令列を基本ブロックに分割した場合の各基本ブロックの先頭となる仮想マシン命令であるか否かを示すブロック先頭情報の入力を受け付けるブロック先頭情報獲得手段と、前記仮想マシン命令列を構成する仮想マシン命令それぞれについて、対応する実マシン命令列に変換する変換手段と、前記変換手段によって得られた実マシン命令列における実マシン命令であって、前記ブロック先頭情報獲得手段が受け付けたブロック先頭情報によって特定された基本ブロックの先頭となる仮想マシン命令に対応する実マシン命令が、前記実マシンによってアドレシングされ得ない位置に配置される命令であるか否かを判定する分岐違反判定手段と、前記分岐違反判定手段によって前記実マシン命令が前記位置に配置される命令であると判定された場合に、前記実マシン命令が前記位置に配置されないよう１個以上の無動作命令を前記実マシン命令列に追加挿入して出力する出力手段とを備えることを特徴とする。

【００６１】

【発明の実施の形態】

以下、本発明の実施の形態について、図面を用いて説明する。

（実施形態１）

まず、真のデータ依存関係の影響を吸収する実施形態１に係る仮想マシンシステムについて説明する。

【００６２】

図１は、本実施形態に係る仮想マシンシステムが動作するコンピュータのシステム（ハードウエア）構成図である。このコンピュータ２００は、実マシン２０１、メモリ２０２、キーボード２０３、マウス２０４、ディスプレイ２０６、ハードディスク２０７、ネットワークカード２０８及びそれらを接続する内部バス２０５ａ～２０５ｃから構成され、このハードウェア構成自体は、汎用のパーソナルコンピュータと異ならない。

【００６３】

本実施形態に係る仮想マシン及びその仮想マシンコンパイラは、実マシン２０１用の命令で記述された実行形式のプログラムであり、ハードディスク２０７に

格納されているが、操作者や他のプログラムからの指示によってメモリ２０２にロードされ、実マシン２０１によって実行される。なお、実マシン２０１は、従来技術での説明と同様に、図７５に示された実マシン命令を解読・実行するCPUである。

（仮想マシンの構成）

図２は、本実施形態に係る仮想マシン１００の構成を示す機能ブロック図であり、従来技術の説明における図７１に対応する。この仮想マシン１００は、先行命令情報格納部１０１、命令格納部１０２、デコード部１０３、実行部１１０及びスタック１２０から構成される。

【００６４】

命令格納部１０２は、実行対象となる仮想マシンプログラムを保持する記憶領域であり、一方、先行命令情報格納部１０１は、その仮想マシンプログラムを構成する各仮想マシン命令に対応する先行命令情報を保持する記憶領域である。ここで、先行命令情報とは、仮想マシンプログラムにおいて対象とする仮想マシン命令の次に実行される仮想マシン命令がスタック１２０の階層を増加させる命令であるか減少させる命令であるかを示す１ビットの情報（前者の場合の情報を「U」、後者の場合の情報を「D」と記す。）をいい、後述する本実施形態に係る仮想マシンコンパイラによってソースプログラムから仮想マシンプログラムと共に生成され得る。

【００６５】

図３（ａ）及び（ｂ）は、それぞれ先行命令情報格納部１０１及び命令格納部１０２に格納されている先行命令情報及び対応する仮想マシンコードの例を示す。ここでは、図８０（ａ）に示された仮想マシンプログラムと同一内容、即ち、「2*(3+4)」を計算する仮想マシンプログラム及びその先行命令情報が格納されている。例えば、先行命令情報格納部１０１のアドレス１及び２には、命令格納部１０２の対応するアドレス１及び２に置かれた仮想マシン命令"Push 2"の次に実行される仮想マシン命令"Push 3"がスタック１２０の階層を増加させる旨の先行命令情報「U」が格納されている。

【００６６】

デコード部１０３は、先行命令情報格納部１０１に格納された先行命令情報を参照しながら命令格納部１０２に格納された仮想マシン命令を逐一に読み出して解読し、その結果を実行部１１０に出力するものであり、さらに、先行命令情報読込み部１０４、命令読込み部１０５、検索部１０６、プログラムカウンタ（ＰＣ）１０７及びデコードテーブル１０８から構成される。

【００６７】

ＰＣ１０７は、命令格納部１０２及び先行命令情報格納部１０１それぞれに格納された次に読み出すべき１対の仮想マシン命令及び先行命令情報のアドレス（これらのアドレスは、本実施形態では同一としている。）を保持するための記憶領域であり、実行部１１０によってそのアドレスが書き込まれることによって更新される。なお、ＰＣ１０７は、従来技術と同様に、物理的には、実マシン２０１のレジスタ２番（ｒ２）に割り当てられている。

【００６８】

命令読込み部１０５は、ＰＣ１０７が示す命令格納部１０２のアドレスに置かれた仮想マシン命令を読み出して検索部１０６に送る。同様にして、先行命令情報読込み部１０４は、命令読込み部１０５に同期して動作し、ＰＣ１０７が示す先行命令情報格納部１０１のアドレスに格納された先行命令情報を読み出し、検索部１０６に送る。

【００６９】

デコードテーブル１０８は、この仮想マシン１００が解読・実行する全ての仮想マシン命令（ここでは、図７２に示された仮想マシン命令とする。）のオペコードと上述した２種類の先行命令情報との全ての組み合わせについて、マイクロプログラム記憶部１１１に格納された対応するマイクロプログラムへのジャンプアドレス及びその仮想マシン命令に伴うオペランドの数を予め記憶する。なお、全てのオペコードは１バイト長であり、オペランドの単位も１バイトである点は、従来技術と同じとする。

【００７０】

図４は、デコードテーブル１０８の内容を示す図であり、図７３に示された従来技術におけるデコードテーブル４４０６に対応する。このデコードテーブル１

０８には、従来のデコードテーブル４４０６と相違し、１種類の仮想マシン命令１０８ａにつき、「U」と「D」の２種類の先行命令情報１０８ｂに対応するジャンプアドレス１０８ｃ及びオペランド数１０８ｄが格納されている。例えば、１種類の仮想マシン命令"Push"について、先行命令情報１０８ｂが「U」の場合には、上向きPushを処理するマイクロプログラムへのジャンプアドレスが格納され、一方、先行命令情報１０８ｂが「D」の場合には、下向きPushを処理するマイクロプログラムへのジャンプアドレスが格納されている。

【００７１】

検索部１０６は、命令読込み部１０５から送られてきた仮想マシン命令のオペコードと先行命令情報読込み部１０４から送られてきた先行命令情報との組み合わせに対応するデコードテーブル１０８内のエントリを特定し、そのエントリに格納されたジャンプアドレスを読み出し、それをデコードデータとして実行部１１０に出力する。

【００７２】

実行部１１０は、検索部１０６から送られてきたデコードデータに従って、仮想マシン命令のオペレーションを実行するものであり、さらに、マイクロプログラム記憶部１１１及びスタックポインタ（ＳＰ）１１２を有する。

マイクロプログラム記憶部４４１１は、この仮想マシン１００が解読・実行する全ての仮想マシン命令と上述した２種類の先行命令情報との全ての組み合わせに対応するマイクロプログラムを予め記憶する。これらマイクロプログラムの詳細については後述する。

【００７３】

ＳＰ１１２は、従来技術と同様に、スタック１２０の最上位のアドレスを保持する記憶領域であり、物理的には、実マシン２０１のレジスタ３番（ｒ３）に割り当てられている。

スタック１２０は、実行部１１０による仮想マシンプログラムの実行のための一時的な後入れ先出し方式の記憶領域であり、さらに、TOS変数１２１、Second Of Stack（SOS）変数１２２及びメモリスタック１２３から構成される。TOS変数１２１は、スタック１２０の最上位の値を格納する記憶域であり、物理的には、

実マシン２０１のレジスタ０番（ｒ０）に割り当てられている。SOS変数１２２は、スタック１２０の最上位から２番目の値を格納する記憶域であり、物理的には、実マシン２０１のレジスタ４番（ｒ４）に割り当てられている。メモリスタック１２３は、スタック１２０の最上位から３番目以降の値を格納する記憶域であり、物理的には、メモリ２０２に割り当てられている。

（マイクロプログラム記憶部１１１の内容）

図５（ａ）及び（ｂ）は、それぞれ、マイクロプログラム記憶部１１１に格納されているマイクロプログラムのうち、上向き仮想マシン命令"Push"及び下向き仮想マシン命令"Push"に対応するもののリストを示す。同様に、図６（ａ）及び（ｂ）は、それぞれ、上向き仮想マシン命令"Add"及び下向き仮想マシン命令"Add"に対応するもの、図７（ａ）及び（ｂ）は、それぞれ、上向き仮想マシン命令"Mult"及び下向き仮想マシン命令"Mult"に対応するもののリストを示す。そして、図８（ａ）及び（ｂ）それぞれは、図６（ａ）及び図７（ａ）に示された上向き仮想マシン命令のマイクロプログラムの後半に相当する部分、図６（ｂ）及び図７（ｂ）に示された下向き仮想マシン命令のマイクロプログラムの後半に相当する部分のリストである。なお、ここで用いられている実マシン命令の意味は、図７５に示される通りである。

【００７４】

これらマイクロプログラムのリストと図７４（ａ）～（ｄ）及び図８２（ａ）～（ｄ）に示された従来技術のものと比較して分かるように、本仮想マシン１００のマイクロプログラム記憶部１１１に格納されたマイクロプログラムには以下の特徴がある。

つまり、第１の特徴は、同一の仮想マシン命令であっても先行命令情報が異なれば対応するマイクロプログラムの内容が異なることである。これは、直後に実行される仮想マシン命令によるスタック操作を予め考慮することによって、無駄なスタック操作や、真のデータ依存関係に基づくパイプラインの乱れを回避するためである。例えば、図５（ｂ）に示されるように、仮想マシン命令"Push"のマイクロプログラムであるにも拘わらず、SOS変数１２２の値をメモリスタック１２３にプッシュする操作が記述されていないのは、この仮想マシン命令の先行命

令情報が下向き、即ち、次に実行される仮想マシン命令によってポップ操作が行われるので、無駄となってしまうことを避けるためである。

【００７５】

第２の特徴は、スタック１２０の最上位の記憶域（TOS変数１２１）だけでなく上位から２番目の記憶域（SOS変数１２２）についても、メモリ２０２ではなくレジスタに割り当てられていることである。これは、加算などの演算の対象となる２つの値をレジスタに保持することで、実マシン２０１とメモリ２０２間のデータ転送の回数を削減するためである。例えば、図６（ａ）に示されるように、加算だけのオペレーションであれば、レジスタとメモリ２０２間でのデータ転送は不要となる。

【００７６】

図９は、本仮想マシン１００によって実行される仮想マシン命令が属する種別の変化を示す状態遷移図である。

ここで、各状態は、本仮想マシン１００によって解読・実行される全ての仮想マシン命令とその先行命令情報との組み合わせの全てについて、スタック１２０に対する操作内容の観点から分類した場合における各分類種別に相当する。各状態中に記された「Ｘ、Ｙ（Ｚ）」におけるＸ、Ｙ、Ｚは、それぞれ、演算対象となるスタック格納値の数、演算後に増加するスタックの階層数、先行命令情報を示す。例えば、状態"2,-1(U)"は、２つの値を用いて演算し、その演算後にはスタックの階層数が１つ減少し、先行命令情報「Ｕ」が対応付けられた仮想マシン命令の全てを示す。具体的には、上向き仮想マシン命令"Add"などが該当する。なお、各状態の右下に記された式は、その状態に属する仮想マシン命令によるTOS変数１２１とSOS変数１２２の変化を示し、Xはオペランドを意味する。

【００７７】

また、図９において、遷移矢印線の種別は、遷移元の状態に属する仮想マシン命令に必要なオペレーションのうち、遷移先に依存することなく先行して行うことが可能な処理の内容を示す。従って、各状態から出ていく遷移矢印線の種別は全て同一である。例えば、状態"2,-1(D)"からの遷移先としては、"2,-1(U)","2,-1(D)","1,0(U)","1,0(D)","1,-1(U)","1,-1(D)"の６種類が考えられるが、いず

れに遷移する場合であっても、その遷移矢印線の種別が示す処理"Pop SOS"、即ち、メモリスタック１２３の最上位の値をSOS変数１２２にポップすることが可能であることが分かる。なお、無条件分岐命令"Br"及び終了命令"Stop"は、TOS変数１２１に対して空の演算を行う命令"1,0(U)"（又は"1,0(D)"）としている。

【００７８】

このように、図９に示された状態遷移図は、本仮想マシン１００に係る全ての仮想マシン命令について先行して行うことが保証された処理内容を示す解析結果と言える。そして、この解析結果は、対応するマイクロプログラムに反映されている。つまり、マイクロプログラム記憶部１１１に格納された各マイクロプログラムには、この状態遷移図に示された対応する先行処理（対応する状態から出ていく遷移矢印線が示す処理）が組み込まれている。

（仮想マシンの動作）

以上のように構成された仮想マシン１００の動作について説明する。

【００７９】

図１０は、本仮想マシン１００のデコード部１０３の動作を示すフローチャートであり、従来技術における図７６に対応する。これらの図を比較して分かるように、このデコード部１０３の動作は、従来のデコード部４４０２と基本的な流れは同一であるが、先行命令情報読込み部１０４が命令読込み部１０５と同期して先行命令情報格納部１０１から先行命令情報を読み込む処理（ステップ４９０６）が追加されている点と、テーブル検索（ステップ４９０７）における詳細な動作内容において、従来と異なる。

【００８０】

図１１及び図１２は、それぞれ、図１０に示されたテーブル検索（ステップ４９０７）の詳細を示すフローチャートの前半部及び後半部であり、従来技術における図７７に対応する

。これらの図を比較して分かるように、本実施形態におけるテーブル検索は、従来のものと基本的な流れは同一であるが、本仮想マシン１００の検索部１０６が命令読込み部１０５から送られてくる仮想マシン命令のオペコードだけでなく先行命令情報読込み部１０４から送られてくる先行命令情報との組み合わせを考慮

し、それらが一致するデコードテーブル１０８のエントリに格納されたジャンプアドレス１０８ｃとオペランド数１０８ｄを参照してデコーデータを特定し実行部１１０に出力する処理（ステップ５００３、５００７）が追加されている点で、従来と異なる。

【００８１】

図１３は、図３（ａ）及び（ｂ）に示される先行命令情報及び仮想マシン命令が先行命令情報格納部１０１及び命令格納部１０２に格納されている場合に、それらに基づいてデコード部１０３が実行部１１０に順次出力するデコードデータを示し、従来技術における図８０（ｃ）に対応する。先行命令情報と仮想マシン命令との組み合わせに対応するマイクロプログラムへのジャンプアドレスが出力されていることが分かる。

【００８２】

なお、実行部１１０の基本的な動作手順は、図７９に示された従来技術に係るフローチャートに示される通りである。つまり、実行部１１０は、ＰＣ１０７とＳＰ１１２とを初期化しておいた後に（ステップ４７０２）、デコード部１０３から送られてくるデコードデータを読み込んで（ステップ４７０４）、そのデコードデコードに含まれるジャンプアドレスが示すマイクロプログラム記憶部１１１内のマイクロプログラムに分岐し実行する動作（ステップ４７０５）を繰り返す（ステップ４７０３～４７０４）。

【００８３】

図１４（ａ）及び（ｂ）は、それぞれ、図１３に示されたデコードデータ列に従って実行部１１０が図３（ｂ）に示された仮想マシンプログラムを実行した場合における各仮想マシン命令が実行される前後でのＰＣ１０７、ＳＰ１１２、TOS変数１２１、SOS変数１２２及びメモリスタック１２３の一連の変化の前半部と後半部を示す図であり、従来技術における図８１（ａ）及び（ｂ）や図８３（ａ）及び（ｂ）に対応する。ここで、本図の遷移矢印の中には、対応する先行命令情報（ＵまたはＤ）と仮想マシン命令との組がそれぞれスラッシュ「／」の左側及び右側に記されている。本図においても、ＰＣが９に達した段階で、計算式"2*(3+4)"の値が算出されていることが分かる。

【００８４】

この図１４（ａ）及び（ｂ）において特徴的なことは、例えば、仮想マシン命令"U/Push 3"の実行後において既にSOS変数１２２の値がメモリスタック１２３の最上位に格納されている点、仮想マシン命令"D/Push 4"の実行後においてＳＰ１１２及びメモリスタック１２３の内容が変化していない点などである。これらは、上述した解析に基づく先行処理が行われていることに起因する。

【００８５】

図１５～図１８は、いずれも、本仮想マシン１００が図３（ｂ）に示された仮想マシンプログラムを実行した場合の実マシン２０１のパイプラインの流れ、より具体的には、アドレス７の下向き仮想マシン命令"Add"のマイクロプログラム（図６（ｂ））の後半部分（図８（ｂ）に示されたジャンプ処理）と、続くアドレス８の下向き仮想マシン命令"Mult"のマイクロプログラム（図７（ｂ））の前半部分（乗算計算処理）におけるパイプラインの流れを示す。但し、図１５～図１８は、それぞれ、実マシン２０１が通常マシンであってメモリ参照を行ってから１クロック後にその参照値を使用することができる場合、実マシン２０１がスーパースカラマシンであって同様の場合、実マシン２０１が通常マシンであってメモリ参照を行ってから２クロック後にその参照値を使用することができる場合、実マシン２０１がスーパースカラマシンであって同様の場合におけるパイプラインの流れを示し、従来技術における図９３～図９６に対応する。

【００８６】

図１５と図９３とを比較して分かるように、パイプラインの乱れは、実マシン命令"Jmp"の実行に起因する１個の命令のキャンセルだけとなり、処理全体は１１クロックで完了する。つまり、実マシン２０１が通常マシンであって、かつ、メモリ参照を行ってから１クロック後にその参照値を使用することができる場合には、本仮想マシン１００は、実行速度において、従来の仮想マシンと異ならない。

【００８７】

また、図１６と図９４とを比較して分かるように、本仮想マシン１００によれば、従来の仮想マシンで生じた第１及び第２の依存関係は吸収され、パイプライ

ンの乱れは、実マシン命令"Jmp"の実行に起因する１個の命令のキャンセルだけとなり、処理全体は９クロックで完了する。つまり、実マシン２０１がスーパースカラマシンであって、かつ、メモリ参照を行ってから１クロック後にその参照値を使用することができる場合、本仮想マシン１００によって、１１クロック要する従来の仮想マシンに比べ、実行速度は２２％だけ高速化される。

【００８８】

これは、下向き仮想マシン命令"Add"のマイクロプログラムにおいて、次に実行すべき仮想マシン命令"Mult"の実行のための先行処理、即ち、メモリスタック１２３からSOS変数１２２へのポップ（Load r4,[r3]）及びＳＰ１１２のデクリメント（Dec r3）を実行しているので、メモリ参照（Load r1,[r2]）から分岐（Jmp r1）までに充分な時間が確保され、パイプラインの乱れが吸収されたからである。

【００８９】

また、図１７と図９５とを比較して分かるように、本仮想マシン１００によれば、同様の理由により、従来の仮想マシンで生じた第１及び第２の依存関係は吸収されるので、パイプラインの乱れは、実マシン命令"Jmp"の実行に起因する１個の命令のキャンセルだけとなり、処理全体は１１クロックで完了する。つまり、実マシン２０１が通常マシンであって、かつ、メモリ参照を行ってから２クロック後にその参照値を使用することができる場合、本仮想マシン１００によって、１３クロック要する従来の仮想マシンに比べ、実行速度は１８％だけ高速化される。

【００９０】

また、図１８と図９６とを比較して分かるように、本仮想マシン１００によれば、同様の理由により、従来の仮想マシンで生じた第１の依存関係に基づくハザードの数が低減され、かつ、第２の依存関係は吸収されるので、パイプラインの乱れは、第１の依存関係に基づく１クロック分のハザードと実マシン命令"Jmp"の実行に起因する１個の命令のキャンセルだけとなり、処理全体は１０クロックで完了する。つまり、実マシン２０１がスーパースカラマシンであって、かつ、メモリ参照を行ってから２クロック後にその参照値を使用することができる場合

、本仮想マシン１００によって、１３クロック要する従来の仮想マシンに比べ、実行速度は３０％だけ高速化される。

【００９１】

以上のように、本実施形態に係る仮想マシン１００は、個々の仮想マシン命令の実行において、対応する先行命令情報を参照することで、真のデータ依存関係をもつ２つの実マシン命令を実行する間に、後続する仮想マシン命令の実行において必要とされるスタック処理を先行して実行する。これによって、真のデータ依存関係に基づくパイプラインの乱れが抑えられ、実行速度が向上する。

（仮想マシンコンパイラの構成）

次に、上記仮想マシン１００を対象とする仮想マシンコンパイラについて説明する。

【００９２】

図１９は、上記仮想マシン１００を対象とする仮想マシンコンパイラ３４００の構成を示す機能ブロック図である。本仮想マシンコンパイラ３４００は、高級言語で記述されたソースプログラム３４０４を入力とし、上記仮想マシン１００固有の仮想マシン命令（図７２に示された仮想マシン命令）からなる命令列３４０５ａと、それら仮想マシン命令に対応する先行命令情報の列３４０５ｂとを生成するクロスコンパイラであり、さらに、命令列変換部３４０２、先行命令情報生成部３４０１及び関連付け部３４０３から構成される。

【００９３】

命令列変換部３４０２は、ネットワークカード２０８やハードディスク２０７等から経路Sを経て入力されるソースプログラム３４０４を構文解析して言語変換することによって、上記仮想マシン１００固有の仮想マシン命令からなる命令列を生成し、その命令列を経路C1及びC3を介して先行命令情報生成部３４０１及び関連付け部３４０３に順次出力する。

【００９４】

先行命令情報生成部３４０１は、命令列変換部３４０２から送られてきた個々の仮想マシン命令に対応する先行命令情報を特定し、経路C2を介して関連付け部３４０３に順次出力する。なお、命令列変換部３４０２及び先行命令情報生成部

３４０１は、対応する仮想マシン命令と先行命令情報との組が同期して関連付け部３４０３に到達するように、それらを出力する際のタイミングを調整している。

【００９５】

関連付け部３４０３は、命令列変換部３４０２から送られてくる仮想マシン命令と先行命令情報生成部３４０１から送られてくる先行命令情報とを組にして関連付けた後に、最終的な出力プログラム３４０５として、それぞれを経路D1及びD2を経てメモリ２０２やハードディスク２０７等に出力する。

図２０及び図２１は、経路Sを経て命令列変換部３４０２に入力されるソースプログラム３４０４のデータ構造を説明するための図であり、それぞれ、ソースプログラム３４０４の命令列（ここでは、"x:=(1+2)*(3+4)"）に対応する木構造及びその木構造を構成する各ノードのデータ構造を示す。各ノードは、ソースプログラム３４０４の命令列を構成する個々の命令に対応し、その種別を示す命令種別５２０１、左部分木へのポインタ５２０２及び右部分木へのポインタ５２０３から構成される。

（仮想マシンコンパイラの動作）

次に、以上のように表現されたソースプログラム３４０４が本仮想マシンコンパイラ３４００に入力された場合の動作について説明する。

【００９６】

図２２は、命令列変換部３４０２の動作手順の概略を示すフローチャートである。命令列変換部３４０２は、木構造で表現されたソースプログラム３４０４の命令列を読み込むと（ステップ５４０２）、まず、その木構造の枝を順次に辿るための計算用スタックを初期化する（ステップ５４０３）。そして、その木構造におけるルートノードへのポインタを変数ptrにセットした後に（ステップ５４０４）、その木構造全体に対応する仮想マシン命令列、即ち、仮想マシンコードを生成し（ステップ５４０５）、バイト単位で先行命令情報生成部３４０１及び関連付け部３４０３に送る。

【００９７】

図２３は、図２２におけるステップ５４０５の詳細を示すフローチャートであ

る。命令列変換部３４０２は、木構造における左部分木に位置するノードに対する変換処理（ステップ５６０３～５６０６）を右部分木に位置するノードに対する変換処理（ステップ５６０７～５６１０）に優先させて、各ノードの命令種別５２０１を数値又はアドレスとしてそのまま出力していくか又は対応する仮想マシンコードに変換するかの処理を繰り返す（ステップ５６１１～５６１３）。なお、全てのノードについて処理を繰り返すために、ステップ５６０５及びＳ５６０９において、この処理（ステップ５６０１～５６１４）自体を再帰的に呼び出している。

【００９８】

図２４は、図２３におけるステップ５６１３の詳細を示すフローチャートであり、図２０に示されるサンプルプログラムに対応するものである。命令列変換部３４０２は、各ノードの命令種別５２０１の値kndによって、対応する４種類の仮想マシン命令"Push","Mult","Pop","Add"のいずれかのオペコードを生成していく（ステップ５９０１～５９０９）。

【００９９】

図２５は、先行命令情報生成部３４０１の動作を示すフローチャートである。先行命令情報生成部３４０１は、命令列変換部３４０２から順次送られてくる仮想マシンコードをバイト単位で受け取り（ステップ５５０２）、その２バイト目以降の仮想マシンコードについて、それがオペランドであるか、仮想マシン命令"Push"のオペコードであるか、他の仮想マシン命令のオペコードであるかを判断し、それぞれに対応する先行命令情報Nextを関連付け部３４０３に出力していく（ステップ５５０３～５５０９）。なお、最後に出力する先行命令情報は、固定的に"U"としている（ステップ５５１０）。

【０１００】

図２６は、関連付け部３４０３の動作を示すフローチャートである。関連付け部３４０３は、直前に処理した仮想マシンコードの先行命令情報を記憶する変数prv及び最終的に生成する先行命令情報及び仮想マシンコードのアドレスAddrを初期化した後に（ステップ６００２）、命令列変換部３４０２から送られてくる仮想マシンコードが無くなるまで次の処理（ステップ６００４～６０１０）を繰

り返す（ステップ６００３）。

【０１０１】

つまり、命令列変換部３４０２及び先行命令情報生成部３４０１からそれぞれ経路C1及びC2を経て、対応する１バイト分の仮想マシンコード及び先行命令情報Nextを受け取った後に（ステップ６００４、６００５）、その先行命令情報Nextが直前と同一である旨を示す"X"であるか否か判断することによって（ステップ６００６）、その仮想マシンコードの先行命令情報nowを決定し（ステップ６００７、６００８）、決定した先行命令情報nowと上記仮想マシンコードとを対にしてアドレスAddrで示されるメモリ２０２等の出力領域における位置に出力する（ステップ６００９、６０１０）。

【０１０２】

このようにして、本仮想マシンコンパイラ３４００によって、高級言語等で記述されたソースプログラム３４０４から、本実施形態に係る仮想マシン１００をターゲットとする仮想マシンプログラム、即ち、図２に示された仮想マシン１００の先行命令情報格納部１０１及び命令格納部１０２に格納すべき先行命令情報列及び仮想マシン命令列の組が生成される。

【０１０３】

また、本実施形態に係る仮想マシンコンパイラ３４００は、木構造で表現されたソースプログラム３４０４を入力としたが、本発明はこれに限定されるものではなく、Ｃ言語等のプログラミング言語で記述されたテキストそのものを入力としてもよい。そして、命令列変換部３４０２が、そのテキストを木構造や３オペランド方式等による中間コードに変換する前処理を行ってもよい。

（実施形態２）

次に、割込み処理に基づく実行効率の低下を回避した実施形態２に係る仮想マシンについて説明する。

（仮想マシンの構成）

図２７は、本実施形態に係る仮想マシン３５００の構成を示す機能ブロック図である。本仮想マシン３５００は、命令格納部４４０１、デコード部３５０２、割込み制御部３５１０、実行部４４１０及びスタック４４２０から構成される。

【０１０４】

本図と図７１とを比較して分かるように、本仮想マシン３５００は、従来の仮想マシン４４００とほぼ同一の構成要素を備える。同一の構成要素については、同一の符号を付し、その説明を省略する。

相違する点は、従来の仮想マシン４４００が備える構成要素に加えて、さらに、この仮想マシン３５００への割込み要求に応じた処理を実行制御するための割込み制御部３５１０を備えること、デコード部３５０２からの制御信号及びデコードデータは分岐命令検出部３５０５に出力されていること、実行部３５１５のマイクロプログラム記憶部４４１１には割込み処理のための実マシンプログラムである割込み処理プログラム３５１６が追加して格納されていることである。以下、従来の仮想マシン４４００と相違する点を中心に説明する。

【０１０５】

割込み制御部３５１０は、本仮想マシン３５００が分岐命令を解読・実行するたびに割込み発生の有無を検出し、必要な割込み処理を実行部４４１０に行わせるよう制御するものであり、さらに、分岐命令検出部３５０５、割り込み命令挿入部３５０６及び割込み状態格納部３５０７から構成される。

分岐命令検出部３５０５は、検索部４４０５から信号線D1を介して送られてきたデコードデータが分岐命令"Br","Brz","Brnz","Call","Ret"のいずれかのマイクロプログラムへのジャンプアドレスであるか否か判断し、肯定的に判断した場合にのみ、信号線C2をオンにしてそのデコードデータを割り込み命令挿入部３５０６に出力し、否定的に判断した場合には、信号線C2をオフにしたままそのデコードデータを割り込み命令挿入部３５０６に出力する。

【０１０６】

割込み状態格納部３５０７は、本仮想マシン３５００に対する割込み要求が発生しているか否か及び発生している場合の割込み種別を特定する状態変数IDを保持する記憶域であり、物理的には、メモリ２０２やネットワークカード２０８内のレジスタ等に割り当てられる。

割り込み命令挿入部３５０６は、分岐命令検出部３５０５が分岐命令を検出した旨の通知を信号線C2を介して受けると、その時点における割込み状態格納部３

５０７の状態変数IDを参照することで、割込みが発生しているか検査し、発生している場合に、その分岐命令のデコードデータの出力に先立ち、割込み処理を実行させるためのデコードデータ、即ち、マイクロプログラム記憶部４４１１内の割込み処理プログラム３５１６へのジャンプアドレスと上記状態変数IDとを実行部４４１０に出力する。

【０１０７】

割込み処理プログラム３５１６は、割り込み命令挿入部３５０６から通知された状態変数IDに基づくメモリ２０２上のアドレスに格納された割込みベクタを読み出し、その割り込みベクタが示すジャンプ先のサブルーチンを処理するという割込み処理を行うための実マシンプログラムである。

（仮想マシンの動作）

以上のように構成された仮想マシン３５００の動作について説明する。

【０１０８】

図２８は、デコード部３５０２の動作のうち、テーブル検索とデコードデータの出力処理の詳細を示すフローチャートであり、従来における図７７に対応する。図７７と異なる点は、デコードデータの出力先の変更に伴う処理（ステップ６１０８～６１１１）である。つまり、検索部４４０５は、読込み部４４０３から送られてきた仮想マシン命令のオペコードに対応するジャンプアドレスをデコードテーブル４４０６から読み出し終えると（ステップ６１０６）、分岐命令検出部３５０５に対して、信号線C1をオンにしたまま信号線D1を介してそのジャンプアドレスをデコードデータとして出力する（ステップ６１０８～６１１０）。

【０１０９】

図２９は、分岐命令検出部３５０５の動作を示すフローチャートである。分岐命令検出部３５０５は、信号線D1を介して入力されたデコードデータを読み込んで一時記憶した後に（ステップ６２０２、６２０３）、そのデコードデータddataがマイクロプログラムへのジャンプアドレスであるか否かを信号線C1の状態によって判断する（ステップ６２０４）。

【０１１０】

その結果、そのようなジャンプアドレスである場合にはさらに、そのジャンプ

アドレスが予め内部に記憶する分岐命令"Br","Brz","Brnz","Call","Ret"のマイクロプログラムのジャンプアドレスのいずれかと一致するか否かを判断し（ステップ６２０５）、一致するときにのみ信号線C2をオンにして（ステップ６２０６）、一時記憶していたデコードデータddataを出力する（ステップ６２０６～６２０８）。

【０１１１】

一方、他の場合においては（ステップ６２０４、６２０５）、信号線C2をオフにしたまま、デコードデータddataを出力する（ステップ６２０７、６２０８）。

図３０は、割込み命令挿入部３５０６の動作を示すフローチャートである。割り込み命令挿入部３５０６は、信号線D2を介して入力されたデコードデータを読み込んで一時記憶した後に（ステップ６３０２、６３０３）、そのデコードデータddata2が分岐命令のマイクロプログラムへのジャンプアドレスであるか否か信号線C2の状態によって判断する（ステップ６３０４）。

【０１１２】

その結果、そのようなジャンプアドレスである場合には、割込み状態格納部３５０７に格納された状態変数IDを読み出し（ステップ６３０５）、その値から割込みが発生しているか否か判断し（ステップ６３０６）、発生している場合にのみ、予め定められた割込み処理を実行させるためのデコードデータ（状態変数IDと割込み処理プログラム３５１６へのジャンプアドレス）を実行部４４１０に出力した後に（ステップ６３０７）、一時記憶していた分岐命令のデコードデータddata2を実行部４４１０に出力する（ステップ６３０８）。その結果、実行部３５１５は、分岐命令の実行に先立ち、状態変数IDに基づく割込み処理プログラム３５１６を実行する。

【０１１３】

一方、信号線D2を介して入力されたデコードデータが分岐命令のマイクロプログラムへのジャンプアドレスでないと判断されたり（ステップ６３０４）、割込みが発生していないと判断された（ステップ６３０６）場合には、特別な処理を行なうことなく、単に、一時記憶していた分岐命令のデコードデータddata2を実

行部４４１０に出力する（ステップ６３０８）。

【０１１４】

このようにして、本仮想マシン３５００が１個の分岐命令を解読・実行するたびに、この仮想マシン３５００への割込みが発生しているか否かが検査され、発生している場合には割込み処理が追加されて実行される。

なお、本実施形態に係る仮想マシン３５００によれば、従来の仮想マシン４４００に比べ、１個の分岐命令につき、割込み処理のための１個の分岐命令が追加して実行されるので、メモリアクセスの回数が１回分余計に増える。しかしながら、通常のプログラムでは分岐命令から次の分岐命令までに平均６個の非分岐命令が存在するので、このような仮想マシンプログラムにおいては、本仮想マシン３５００によって増加するメモリアクセスの回数は、１命令当たりに換算すれば、わずか０．２以下である。従って、例えば、本実施形態の割込み処理機能を実施形態１に係る仮想マシン１００に適用することで、TOS変数によるメモリアクセス回数の削減の効果を打ち消してしまうことなく、全体として従来よりもメモリアクセス回数を削減させることができるので、実行速度の高速化に好適な割込み処理機能付き仮想マシンが実現される。

【０１１５】

以上のように、本実施形態に係る仮想マシン３５００によれば、デコード部３５０２と実行部４４１０との間に割込み制御部３５１０が設けられ、分岐命令検出部３５０５によって分岐命令が解読実行される場合に限定して割込み発生の検出と割込み処理が行なわれるので、全ての仮想マシン命令ごとに行われる場合に比べて実行効率の低下が抑止され、適度な頻度で割込み検出処理が行なわれる。

【０１１６】

なお、本実施形態では、デコード部３５０２からのデコードデータを監視することで分岐命令の発生が検出されたが、これをデコード部３５０２に入力される仮想マシン命令（のオペコード）を監視することで同一の検出をしてもよい。

同様に、デコード部３５０２からのデコードデータを監視するのではなく、マイクロプログラム記憶部４４１１に記憶された分岐命令に対応するマイクロプログラムの中に、割り込み命令挿入部３５０６が有する手順を予め追加して記述し

ておく方法であっても同じ結果が得られる。

（実施形態３）

次に、実施形態２と別の方法によって割込み処理に基づく実行効率の低下を回避した実施形態３に係る仮想マシンについて説明する。

（仮想マシンの構成）

図３１は、本実施形態に係る仮想マシン３６００の構成を示す機能ブロック図である。本仮想マシン３６００は、命令格納部４４０１、デコード部３５０２、割込み制御部３６１０、実行部４４１０及びスタック４４２０から構成される。

【０１１７】

本図と図２７とを比較して分かるように、本仮想マシン３５００は、実施形態２に係る仮想マシン３５００とほぼ同一の構成要素を備える。相違する点は、実施形態２における分岐命令検出部３５０５がブロック化部３６０５に置き換えられていること、そのブロック化部３６０５についての接続関係である。以下、実施形態２に係る仮想マシン３５００と相違する点を中心に説明する。

【０１１８】

ブロック化部３６０５は、本仮想マシン３６００により解読された仮想マシンコードのブロック化、即ち、一定個数（例えば、１０バイト）の仮想マシンコードが解読されたか否かを繰り返し検出し、ブロック化（検出）する度にその旨を割り込み命令挿入部３５０６に通知する。

（仮想マシンの動作）

以上のように構成された仮想マシン３６００の動作について説明する。

【０１１９】

図３２は、ブロック化部３６０５の動作を示すフローチャートである。ブロック化部３６０５は、信号線D1を介して入力されたデコードデータを読み込んで一時記憶した後に（ステップ６４０２、６４０３）、その時点でのＰＣ４４０４の値を読み出す（ステップ６４０４）。すなわち、デコード部３５０２から送られてきたデコードデータがいずれのアドレスに置かれた仮想マシンコードに対応するものかを参照する。

【０１２０】

そして、その値PCを予め記憶する定数bsizeで割った余りmを算出し（ステップ６４０５）、その余りmがゼロであるか否かを判断し（ステップ６４０６）、ゼロであるときにのみ信号線C2をオンにして（ステップ６４０７）、一時記憶していたデコードデータddataを出力する（ステップ６４０７～６４０９）。

一方、ゼロでない場合には（ステップ６４０６）、信号線C2をオフにしたまま、デコードデータddataを出力する（ステップ６４０８、６４０９）。

【０１２１】

割り込み命令挿入部３５０６は、実施形態２の場合と同様にして、信号線C2がオンとなると、割込み発生の有無を検査し、発生している場合にのみ、割込み処理を実行させるためのデコードデータ、即ち、マイクロプログラム記憶部４４１１内の割込み処理プログラム３５１６へのジャンプアドレスと上記状態変数IDとを実行部４４１０に出力する。

【０１２２】

このようにして、本仮想マシン３６００が一定個数bsizeの仮想マシンコードが解読される度に、この仮想マシン３６００への割込みが発生しているか否かが検査され、発生している場合には割込み処理が追加されて実行される。つまり、本実施形態に係る仮想マシン３６００によれば、割込み検出処理が実行される頻度は、定数"bsize"で指定される数の仮想マシンコードにつき一回に限定される。

【０１２３】

従って、実施形態２の場合と同様に、定数"bsize"を一定値以上とし、本実施形態の割込み処理機能を実施形態１に係る仮想マシン１００に適用することで、実施形態１に係る仮想マシン１００におけるTOS変数によるメモリアクセス回数の削減の効果を打ち消してしまうことなく、全体として従来よりもメモリアクセス回数を削減させた高速な割込み処理機能付き仮想マシンが実現される。

【０１２４】

なお、本実施形態に係る仮想マシン３６００によれば、ブロック化部３６０５はＰＣ４４０４の値を参照しているが、ＰＣ４４０４自体は実マシン２０１のレジスタ２番（ｒ２）に割り当てられているので、この参照によってはメモリアク

セスの回数が増加することはない。

また、本実施形態に係る仮想マシン３６００によれば、定数"bsize"を変更するだけで割込み検出処理の実行頻度を調整することが可能となり、メモリアクセスの回数を柔軟に制御できる仮想マシンが実現される。

【０１２５】

なお、本実施形態のブロック化部３６０５は、デコード部３５０２から送られてきたデコードデータに対応するＰＣ４４０４の値と定数bsizeを比較したが、これに代えて、内部にカウンタを設けることによって、デコード部３５０２からの信号線C1がオンになった回数をカウントし、その値と定数bsizeとを比較してもよい。これによって、一定バイト数の仮想マシンコードではなく、一定個数の仮想マシン命令ごとに割込み検出処理が行なわれることになる。

【０１２６】

さらに、本実施形態では、割込み制御部３６１０が独立してブロック化を行ったが、実行部４４１０がＰＣ４４０４を参照できることを考慮すれば、マイクロプログラム記憶部４４１１の内容として割込み制御部３６１０の処理手順を予め追加しておく方法であっても同じ結果が得られる。

（実施形態４）

次に、実マシンのアーキテクチャへの非依存性を高めた実施形態４に係る仮想マシンについて説明する。

（仮想マシンの構成）

図３３は、本実施形態に係る仮想マシン３７００の構成を示す機能ブロック図である。本仮想マシン３７００は、命令格納部３７０１、デコード部４４０２、実行部３７１０及びスタック４４２０から構成される。

【０１２７】

本図と図７１とを比較して分かるように、本仮想マシン３７００は、従来の仮想マシン４４００とほぼ同一の構成要素を備える。相違する点は、命令格納部３７０１に格納されている内容と、実行部３７１０に領域判定部３７０４とアドレス変換部３７０５が加えられていることと、実マシン関数記憶部３７０６が追加されていることである。以下、従来の仮想マシン４４００と相違する点を中心に

説明する。

【０１２８】

実マシン関数記憶部３７０６は、実マシン命令で記述された関数（以下、「実マシン関数」と呼ぶ。）の集まり、つまり、仮想マシンプログラムで必要とされる定型的な処理を行う関数の集合を実行時ライブラリとして予め記憶しており、物理的には、メモリ２０２に割り当てられている。具体的には、後述する第０～（ＲＭmax－ＲＭmin）番までの合計（ＲＭmax－ＲＭmin＋１）個の実マシン関数を記憶している。

【０１２９】

命令格納部３７０１は、実行対象となる仮想マシンプログラムだけでなく、実マシン関数テーブルも予め保持している。ここで、実マシン関数テーブルとは、上記実マシン関数記憶部３７０６に格納された各実マシン関数それぞれへのポインタ（先頭アドレス）の集まりである。

図３４は、命令格納部３７０１のメモリマップ、即ち、仮想マシン３７００から見たメモリ領域の用途分割を示す。アドレスＶＭmin～ＶＭmaxの領域は、仮想マシンプログラム６５０１、即ち、仮想マシン命令で記述された関数（仮想マシン関数）の集まりが配置される領域に割り当てられ、これに続くアドレスＲＭmin～ＲＭmaxの領域は、実マシン関数テーブル６５０２が配置される領域に割り当てられている。なお、実マシン関数テーブル６５０２の領域は、仮想マシンプログラム６５０１の領域の直後に置かれている。つまり、アドレスＲＭminは、アドレスＶＭmax＋１に等しい。

【０１３０】

図３５は、図３４に示された実マシン関数テーブル６５０２の構造を示す。命令格納部３７０１のアドレスＲＭmin～ＲＭmaxの領域には、各番地ごとに、第０～（ＲＭmax－ＲＭmin）番の実マシン関数へのポインタが格納されている。但し、各ポインタは、アドレスの逆順に対応づけて格納されている。例えば、第０番の実マシン関数は、アドレスＲＭmaxに置かれた仮想マシン関数が呼び出された場合に実行される関数であり、同様に、第（ＲＭmax－ＲＭmin）番の実マシン関数は、アドレスＲＭminに置かれた仮想マシン関数が呼び出された場合に実行さ

れる関数である。

【０１３１】

領域判定部３７０４は、デコード部４４０２から出力されたデコードデータを監視することで、実行部３７１０による実行対象が関数呼び出し命令"Call"である場合に、その命令の実行に先立ち、その呼び出し先が仮想マシンプログラム６５０１内であるか実マシン関数テーブル６５０２が置かれた領域内であるかを判定する。

【０１３２】

アドレス変換部３７０５は、領域判定部３７０４によって実行対象の仮想マシン命令が実マシン関数テーブル６５０２の領域内への関数呼び出し命令"Call"であると判定された場合にのみ、その仮想マシン命令"Call"の実行に代えて、そのジャンプアドレスに対応する実マシン関数テーブル６５０２内の関数ポインタが示す実マシン関数記憶部３７０６内の実マシン関数を直接実マシン２０１に実行させる。

（仮想マシンの動作）

以上のように構成された仮想マシン３７００の動作について説明する。

【０１３３】

図３６は、本仮想マシン３７００の実行部３７１０の動作、特に、デコード部４４０２から関数呼び出し命令"Call"のデコードデータが送られてきた場合の領域判定部３７０４とアドレス変換部３７０５の動作を中心としたフローチャートである。

領域判定部３７０４は、信号線Ｒの状態及び検索部４４０５からのデコードデータを監視することによって、デコード部４４０２から関数呼び出し命令"Call"のオペランドが送られてきたことを知ると、その仮想マシン命令の実行に先立ち、そのオペランドが示す呼び出し先JaddrがアドレスRM_{min}～RM_{max}の範囲、即ち、実マシン関数テーブル６５０２が置かれた領域内であるかを判定する（ステップ６８０２～６８０４）。

【０１３４】

その結果、その呼び出し先Jaddrが領域内であると判定された場合には、アド

レス変換部３７０５は、上記逆順に基づいて、呼び出し先Jaddrに対応する実マシン関数テーブル６５０２へのインデックスidxを算出し（ステップ６８０５）、そのインデックスidxが示す実マシン関数テーブル６５０２のエントリに格納されたポインタptrを読み出す（ステップ６８０６）。そして、実行部３７１０は、元の仮想マシン命令"Call"に代えて、ポインタptrが示す実マシン関数記憶部３７０６内の実マシン関数を直接実行する（ステップ６８０７）。

【０１３５】

一方、領域判定部３７０４によって関数呼び出し命令"Call"の呼び出し先Jaddrが実マシン関数テーブル６５０２の領域外であると判定された場合には、実行部３７１０は、通常の関数呼び出し処理を続行する（ステップ６８０８～６８１０）。つまり、実行部３７１０は、戻り番地を記憶した後に（ステップ６８０８、６８０９）、その呼び出し先Jaddrに置かれた仮想マシン関数を実行する（ステップ６８１０）。

【０１３６】

このようにして、仮想マシン命令"Call"の呼び出し先Jaddrが仮想マシンプログラム６５０１の領域に属する場合には、そのまま仮想マシン関数が呼び出され、一方、実マシン関数テーブル６５０２の領域に属する場合には、対応する実マシン関数が実行される。

ここで、図３４に示されたメモリマップから分かるように、関数呼び出し命令"Call"に対して仮想マシン関数を実行させるか実マシン関数を実行させるかは、２つの領域６５０１、６５０２を区切る境界線を移動させることによって容易に変更することができる。例えば、境界線となるアドレスVMmaxを下げた場合には、これに伴ってアドレスRMminも下がるので、たとえ同一アドレスJaddrへの関数呼び出し命令"Call"であっても、それまでの仮想マシン関数の実行に代えて実マシン関数を実行させることが可能となる。同様に、境界線のアドレスVMmaxを上げた場合には、それまでの実マシン関数の実行に代えて仮想マシン関数を実行させることが可能となる。

【０１３７】

以上のように、本実施形態に係る仮想マシン３７００によれば、わずか１つの

パラメータＶＭmaxの値を変更することによって、仮想マシン関数への呼び出しをそのまま実行させたり実マシン関数に置き換えて実行させたりする制御が可能となる。従って、本仮想マシン３７００は、複数の異なる種別の実マシンやコンピュータ環境の下で動作させる仮想マシンとして好適なアーキテクチャを備えていると言える。複数の異なるアーキテクチャの実マシンやコンピュータそれぞれに応じて仮想マシンプログラムの一部を実マシン関数に置き換えておく場合には、それらアーキテクチャごとに仮想マシン関数として実行させる箇所と実マシン関数として実行させる箇所とを容易に切り分けられるからである。

【０１３８】

これによって、実行速度を低下させることなく、実マシンのアーキテクチャへの非依存性を高めた仮想マシンが実現される。

なお、本実施形態では、命令格納部３７０１のアドレスＶＭmin～ＶＭmaxの領域には、仮想マシンプログラム６５０１だけが配置されていたが、本発明はこれに限られず、例えば、図３７に示されるように、各アドレスごとのメモリ属性６７０１（Ｖ又はＲ）と、その属性に対応するデータ（仮想マシンプログラム）又は実マシン関数テーブルへのインデックスを格納してもよい。これによって、境界線ＶＭmaxを移動させることなく、同一アドレスへの仮想マシン関数の呼び出しに対してそのまま仮想マシン関数として実行させるか実マシン関数を直接実行させるかの切り替えが可能となる。

（実施形態５）

次に、仮想マシンプログラムのキャッシュブロック化やその場式コンパイラのコンパイル時間が短縮化される実施形態５に係る仮想マシンシステムについて説明する。

（仮想マシンの構成）

図３８は、本実施形態に係る仮想マシン３８００の構成を示す機能ブロック図である。本仮想マシン３８００は、命令格納部３８０１、デコード部３８０２、実行部３８１０及びスタック４４２０から構成される。

【０１３９】

本図と図７１とを比較して分かるように、本仮想マシン３８００は、従来の仮

想マシン４４００とほぼ同一の構成要素を備える。相違する点は、命令格納部３８０１に格納されている内容と、ＰＣ３８０４の構成と、実行部３８１０に分岐先変換部３８１１が追加されていることである。以下、従来の仮想マシン４４００と相違する点を中心に説明する。

【０１４０】

命令格納部３８０１は、実行対象となる仮想マシンプログラムを命令ブロックと呼ばれる単位に区分して記憶しているものであり、各命令ブロックを記憶する複数の命令ブロック格納領域３８５２ａ～３８５２ｄからなる。

ここで、命令ブロックとは、本仮想マシンプログラムを基本ブロックに分割し、得られた基本ブロックそれぞれにユニークな識別子を付与し、それら基本ブロックを論理的に接続するための分岐命令を後続位置に追加したものをいい、後述する本仮想マシン３８００固有のコンパイラによって生成される。なお、基本ブロックとは、先頭に配置された命令をそのブロックへの唯一の入口とし、最後部に配置された命令をそのブロックからの唯一の出口とする命令列をいう。また、本実施形態では、識別子は、命令ブロック格納領域における各命令ブロックの先頭位置を特定するアドレス情報からなる。

【０１４１】

命令ブロック格納領域３８５２ａ～３８５２ｄそれぞれは、各命令ブロックの識別子を格納している識別子格納領域３８５３ａと、各命令ブロックに属する仮想マシン命令のうち分岐命令を除くもの（以下、「非分岐命令」と呼ぶ。）だけを格納している非分岐命令格納領域３８５４ａと、その分岐命令だけを格納している分岐命令格納領域３８５５ａとからなる。

【０１４２】

図３９は、命令格納部３８０１に格納されている仮想マシンプログラムの格納状態の例を示す図であり、図９７に示されたサンプル仮想マシンプログラムが格納されている場合に相当する。

本図に示されるように、上記仮想マシンプログラムは５個の命令ブロック３８５２ａ～３８５２ｄに分割され、各命令ブロック３８５２ａ～３８５２ｄは、その命令ブロックの識別子３８５３ａ～３８５３ｄと、各命令ブロックに属する基

本ブロックの非分岐命令だけからなる部分３８５４ａ～３８５４ｄと、その基本ブロックの最後部に配置された分岐命令及び次の命令ブロックに接続するための分岐命令とからなる部分３８５５ａ～３８５５ｄとから構成される。

【０１４３】

なお、図３９及び図９７それぞれに示された仮想マシンプログラムは、図７２に示された各仮想マシン命令の意味から明らかなように、それらプログラムの制御フローは図４０に示される通りであり、実質的に同一の処理内容を行うことは明らかである。

ＰＣ３８０４は、さらに、識別子セグメントレジスタ３８０４ａとオフセットカウンタ３８０４ｂとから構成される。識別子セグメントレジスタ３８０４ａは、次に読み出すべき命令格納部３８０１中の仮想マシンコードが属する命令ブロックの識別子に相当するセグメントアドレス（以下、「識別子セグメント」と呼ぶ。）を保持し、オフセットカウンタ３８０４ｂは、その仮想マシンコードの命令ブロックにおけるオフセットを保持している。

【０１４４】

なお、本仮想マシン３８００は、図４１に示されるように、１６ビットによるアドレシングを行い、その上位８ビットを識別子セグメント、下位８ビットをオフセットとしている。つまり、識別子セグメントレジスタ３８０４ａには８ビットの識別子セグメントが格納され、オフセットカウンタ３８０４ｂには８ビットのオフセットが格納され、これらが連結された１６ビットのアドレスによって、命令格納部３８０１中の１個の仮想マシンコードが特定される。

【０１４５】

分岐先変換部３８１１は、実行部３８１０において分岐命令を実行する場合に、その分岐命令のオペランド、即ち、分岐先となる命令ブロックの識別子を、識別子セグメントとオフセットとの組み合わせに変換し、それぞれの値をＰＣ３８０４に格納することで更新する。

（仮想マシンの動作）

以上のように構成された仮想マシン３８００の動作について説明する。

【０１４６】

デコード部３８０２及び実行部３８１０は、従来の仮想マシン４４００とほぼ同様の動作をする。異なる点は、通常の順次実行においては、実行部３８１０によってＰＣ３８０４のオフセットカウンタ３８０４ｂだけがインクリメントされる点と、分岐命令の実行においては、分岐先変換部３８１１によってＰＣ３８０４の識別子セグメントレジスタ３８０４ａとオフセットカウンタ３８０４ｂが更新される点である。

【０１４７】

図４２は、実行部３８１０における分岐先変換部３８１１の動作を示すフローチャートである。分岐先変換部３８１１は、デコード部３８０２から送られたきた分岐命令のオペランド、即ち、８ビットで表現された命令ブロックの識別子Jaddrを獲得すると（ステップ８１０２）、それを分岐先の識別子セグメントとすると共にオフセットをゼロとして１６ビットの物理アドレスを生成し、それぞれでＰＣ３８０４の識別子セグメントレジスタ３８０４ａ及びオフセットカウンタ３８０４ｂを更新する（ステップ８１０３）。

【０１４８】

図４３は、図３９に示された仮想マシンプログラムにおける論理的なアドレスと識別子とを物理的なアドレスに置き換えたものであり、分岐先変換部３８１１によるアドレス変換の結果を示す。例えば、図３９における識別子番号１の命令ブロックにおける分岐命令"Brz"のオペランド"x03"は、分岐先変換部３８１１によって識別子番号３の命令ブロックの先頭への物理的な番地"x0300"に変換される。

【０１４９】

以上のようにして、実行部３８１０は、分岐命令を実行するごとに、そのオペランドで指定された命令ブロックの先頭に分岐する制御を行う。このようにして、本仮想マシン３８００は、命令ブロックに分割されて格納されている仮想マシンプログラムに対して、命令ブロックに分割されなかった場合と実質的に同じ処理手順により、解読・実行を行う。

（仮想マシンコンパイラの構成）

次に、本仮想マシン３８００を対象とする仮想マシンコンパイラについて説明

する。

【０１５０】

図４４は、実施形態５に係る仮想マシンコンパイラ７６６０の構成示すブロック図である。本仮想マシンコンパイラ７６６０は、Ｃ言語等の高級言語で記述されたソースプログラム７６５０を入力とし、これを上記仮想マシン３８００の命令格納部３８０１に格納するのに適した形式、つまり、命令ブロック集合７６５１に変換するコンパイラであり、中間命令列変換部７６６１、生成部７６６２及びブロック化部７６６３から構成される。

【０１５１】

中間命令列変換部７６６１は、入力されたソースプログラムを構文解析した後に最適化のために一時的な中間コードに展開し、生成部７６６２は、中間命令列変換部７６６１により展開された中間コードを、例えば図９７に示されるような仮想マシンプログラム７６６４のコードに変換する。これら中間命令列変換部７６６１及び生成部７６６２は、従来の一般的な仮想（又は実マシン）コンパイラが有する機能と異ならない。

【０１５２】

ブロック化部７６６３は、生成部７６６２から生成された仮想マシンプログラム７６６４を上記命令格納部３８０１に格納可能な命令ブロックの集まりに変換するものであり、その際の主な処理は、基本ブロックへの分割と、その分割に伴うアドレス解決である。ここで、アドレス解決とは、仮想マシンプログラム７６６４中の分岐命令で用いられていた分岐先を命令ブロックの識別子IDに置き換える処理である。

【０１５３】

そして、このブロック化部７６６３は、そのアドレス解決のために、一時的な変数テーブルとして、分岐先変更テーブル７６６３ａを作成し用いる。

図４５は、分岐先変更テーブル７６６３ａの構造を示す図である。

本テーブルの各行（エントリ）は、生成される個々の命令ブロック及びブロック化部７６６３に入力された仮想マシンプログラム７６６４中の個々の分岐命令それぞれに対応して作成されるものである。各エントリにおいて、「コード位置

」は、当該命令ブロックの先頭命令又は当該分岐命令の仮想マシンプログラム７６６４上でのアドレス、「登録フラグ」は、当該分岐命令についてのアドレス解決が終了したか否かのフラグ、「参照位置識別子」及び「参照位置offset」は、当該命令ブロックに分岐する分岐命令又は当該分岐命令が置かれている命令ブロックの識別子及びオフセットを示す。

（仮想マシンコンパイラの動作）

図４６は、本仮想マシンコンパイラ７６６０の特徴的な処理、即ち、ブロック化部７６６３の動作を示すフローチャートである。まず、ブロック化部７６６３は、命令ブロック集合７６５１として生成する命令ブロックの識別子IDと、各命令ブロックにおける相対的な命令格納位置を示すポインタoffsetと、仮想マシンプログラム７６６４から順次に読み出す１バイトの仮想マシンコードVCの位置を示すカウンタPCと、分岐先アドレスを変更するための分岐箇所の個数を示すカウンタRcountとを初期化する（ステップ７６０２～７６０３）。

【０１５４】

ブロック化部７６６３は、基本的には、カウンタPCをインクリメントしながら仮想マシンプログラム７６６４から１バイトずつ仮想マシンコードVCを読み出し、その仮想マシンコードVCが属すべき命令ブロックの識別子ID及びその命令ブロック中での相対位置であるポインタoffsetと共に、命令ブロック集合７６５１として出力する（ステップ７６０４～７６１１）。

【０１５５】

このとき、仮想マシンコードVCが基本ブロックの先頭であるか否かの判断（ステップ７６０７）と、分岐命令であるか否かの判断を行い（ステップ７６０８）、それぞれ肯定的に判断された場合には、対応する特別な処理（ステップ７７０１～７７０４及びステップ７６０９）を行う。

図４７は、図４６におけるステップ７６０７、即ち、仮想マシンコードVCが基本ブロックの先頭とすべきものか否かの判断処理の詳細を示す。つまり、仮想マシンコードVCが分岐先の命令及び分岐命令の直後に配置された命令のいずれかに該当する場合には、その仮想マシンコードVCは基本ブロックの先頭と判断する（ステップ７３０２～７３０６）。

【０１５６】

図４６に示されるように、仮想マシンコードVCが基本ブロックの先頭であると判断すると、新たな命令ブロックを生成するために識別子IDを更新すると共に（ステップ７７０１）、直前の命令ブロック（識別子ID)の最後部と次の命令ブロック（識別子NID）の先頭とを接続するための無条件分岐命令を生成する（ステップ７７０２）。そして、この新たな命令ブロックに属する仮想マシンコードを生成するための準備をした後に（ステップ７７０３）、この新たな識別子NIDの付与に伴うアドレス解決を行う（ステップ７７０４）。

【０１５７】

また、ステップ７６０８において、仮想マシンコードVCが分岐命令であると判断した場合には、その分岐先を適切に変更するためのアドレス解決を行う（ステップ７６０９）。このブロック化部７６６３による分割処理と新たな分岐命令の追加処理などによって、元の仮想マシンプログラム７６６４における各仮想マシン命令の配置位置が変動するからである。

【０１５８】

図４８は、図４６におけるステップ７７０４、即ち、新たな命令ブロックの識別子NIDの付与に伴うアドレス解決の詳細を示す。ここでは、ブロック化部７６６３は、この新たな命令ブロックの識別子NIDの付与に伴って初めてアドレス解決が可能な分岐命令が発見された場合には、そのアドレス解決を行い（ステップ７９０５～７９１０）、そうでない場合には、以降の処理においてこの命令ブロックに分岐する分岐命令のアドレス解決を行うために、分岐先変更テーブル７６６３ａへの追加登録をする（ステップ７９１３、７９１４）。

【０１５９】

図４９は、図４６におけるステップ７６０９、即ち、仮想マシンプログラム７６６４中の分岐命令で指定されていた分岐先のアドレス解決の詳細を示す。ここでは、ブロック化部７６６３は、この分岐命令が前方への分岐、即ち、分岐先変更テーブル７６６３ａに既に登録された命令ブロックへの分岐である場合には、その分岐先を命令ブロックの識別子に置き換えることによってアドレス解決を行い（ステップ７８０２～７８０９、７８１２）、そうでない場合には、アドレス

未解決として新たなエントリを分岐先変更テーブル７６６３ａに追加登録する（ステップ７８１０、７８１１）。

【０１６０】

以上のように、仮想マシンコンパイラ７６６０によって、高級言語で書かれたソースプログラムは、一旦、図９７に示されるような一般的な仮想マシンプログラム７６６４に変換された後に、基本ブロックに分割されて識別子が付与され、さらに、各基本ブロックを接続するための分岐命令が追加された後に、識別子付与に伴うアドレス解決が行なわれることで、本実施形態に係る仮想マシン３８００の実行対象となる命令ブロック集合７６５１に変換される。

（考察）

このように、本実施形態に係る仮想マシン３８００及び仮想マシンコンパイラ７６６０によって、実行対象となる仮想マシンプログラムは、図９７に示されるような従来形式のまま命令格納部３８０１に配置され実行されるのではなく、基本ブロックに分割された状態で命令格納部３８０１に配置され実行されることになるが、そのことの技術的意義について考察する。

【０１６１】

まず、その場式コンパイラを採用した場合のコンパイル時間という観点から考察する。

上述したように、従来のその場式コンパイラによれば、仮想マシンプログラムにおける各分岐先について、その位置が一定の制限を違反していないかどうかの解析が必要とされ、違反している場合には、その分岐先を移動させる等の処理が必要とされた。ところが、本仮想マシンシステムによれば、いずれの分岐先も、必ず、各命令ブロックの先頭であることが保証されている。従って、従来において必要とされた、分岐先についての上記処理のほとんどは不要となる。

【０１６２】

また、従来のその場式コンパイラによれば、遅延分岐等のために、分岐命令の後続位置に配置する命令、即ち、分岐の正否の影響を受けない命令を特定する等の処理が必要とされた。ところが、本仮想マシンシステムによれば、命令格納部３８０１に格納された仮想マシンプログラムは、各命令ブロック中において非分

岐命令格納領域と分岐命令格納領域とに分けられており、さらに、分岐命令格納領域においては１個の分岐命令のあとには高々１個の分岐命令だけが後続することが保証されている。従って、従来において必要とされた、遅延分岐等についての上記処理のほとんどは不要となる。

【０１６３】

次に、仮想マシンのキャッシュ機構への対応化という観点から考察する。

従来のようなキャッシュ機構への対応化によれば、仮想マシンプログラムをキャッシュブロックに分割する際に、プログラムカウンタを変更する全ての仮想マシン命令についてキャッシュブロックの境界をまたいで変更していないか判断する必要があった。ところが、本仮想マシンシステムによれば、命令格納部３８０１に格納された仮想マシンプログラムを命令ブロック単位でキャッシュさせるならば、キャッシュの境界をまたいでプログラムカウンタを変更する仮想マシン命令は、分岐命令格納領域３８５５ａに属する分岐命令だけとなる。

【０１６４】

図５０は、本実施形態に係る仮想マシン３８００において、命令ブロック単位でキャッシュさせた場合のＰＣ３８０４、命令ブロック格納領域３８５２ａ～３８５２ｄ及びキャッシュテーブル８４０４の関係を示す図であり、従来における図１０２に対応する。従来では、１０番地分の命令列６９０３がキャッシュブロックとして命令キャッシュ６９０２に置かれたが、本仮想マシン３８００の場合には、命令ブロック３８５２ａ～３８５２ｄの単位で命令キャッシュ８４０２に配置され、それらの識別子がキャッシュテーブル８４０４で管理される様子が示されている。

【０１６５】

図５１は、本実施形態に係る仮想マシン３８００において、命令ブロック単位でキャッシュさせた場合の実行部３８１０による分岐命令の実行処理を示すフローチャートであり、キャッシュ機構に対応させていない場合の図４２に対応する。これらを比較して分かるように、本仮想マシン３８００をキャッシュ機構に対応した仮想マシンとするには、実行部３８１０が、キャッシュテーブル８４０４内の識別子を参照することで命令ブロック単位でキャッシュヒットの有無を判断

し（ステップ８５０４）、ミスヒット時には命令キャッシュ８４０２への読込みを行えばよい（ステップ８５０５）。

【０１６６】

このように、命令ブロック単位で仮想マシンプログラムをキャッシュすることにより、従来において必要とされたキャッシュブロックの境界に伴う判断処理などが不要になる。そして、ミスヒットに伴うキャッシュへの読込みが発生した場合であっても、もともとの仮想マシンプログラムが予め命令ブロック単位で分割格納されているので、読込み時の負荷も削減される。

【０１６７】

以上のように、本実施形態に係る仮想マシンシステムでは、ソースプログラムは一般的な仮想マシンプログラムに変換された後にさらに基本ブロックを単位とする命令ブロックに分割されて命令格納部３８０１に格納され、全ての分岐命令の分岐先は命令ブロックの識別子にて指定されている。これにより、その場式コンパイラでの分岐先命令のアドレス解析処理は単純化され、コンパイル時間が短縮される。また命令キャッシングを命令ブロック単位で行うことで、キャッシュ境界に伴う判定処理は簡略化され、仮想マシンにキャッシュを導入した場合の実行効率の低下は、従来に比較して抑えられる。

【０１６８】

なお、本実施形態の仮想マシンコンパイラ７６６０は、中間命令列変換部７６６１及び生成部７６６２を備えたが、これらに代えて、ソースプログラムから仮想マシンプログラムを生成する一般的なコンパイラを採用してもよいことは言うまでもない。

（実施形態６）

次に、上記実施形態５に係る仮想マシンにおける解読処理の高速化を図った実施形態６に係る仮想マシンについて説明する。

（仮想マシンの構成）

図５２は、本実施形態に係る仮想マシン３９００の構成を示す機能ブロック図である。本仮想マシン３９００は、命令格納部３９０１、デコード部３９０２、実行部３８１０及びスタック４４２０から構成される。

【０１６９】

本図と図３８とを比較して分かるように、本仮想マシン３９００は、実施形態５に係る仮想マシン３８００とほぼ同一の構成要素を備える。相違する点は、命令格納部３９０１に格納されている内容と、デコード部３９０２にカレントフラグ記憶部３９０７が追加されていること、命令読込み部３９０３の機能と、実行部３９１０にカレントフラグ読出制御部３９１２が追加されていることである。以下、実施形態５に係る仮想マシン３８００と相違する点を中心に説明する。

【０１７０】

命令格納部３９０１は、実施形態５の命令格納部３８０１と比較し、予め、実行対象となる仮想マシンプログラムを複数の命令ブロック３９５２ａ～３９５２ｄの単位で記憶している点において共通するが、各命令ブロック格納領域３９５２ａ～３９５２ｄには、当該命令ブロックの非分岐命令格納領域及び分岐命令格納領域（これらを併せて「仮想マシンコード領域」と呼ぶ。）に格納されている全ての仮想マシンコードに対応するデコードデータ列（以下、「デコード命令列」と呼ぶ。）を格納するためのデコード命令列格納領域３９５６ａ～３９５６ｄが設けられている点において異なる。

【０１７１】

図５３（ａ）～（ｃ）は、命令格納部３９０１に格納されている仮想マシンプログラムの格納状態の例を示す図であり、図９７に示されたサンプル仮想マシンプログラムが格納されている場合に相当する。

本図に示されるように、各命令ブロック格納領域３９５２ａ～３９５２ｃに設けられたデコード命令列格納領域３９５６ａ～３９５６ｃは、さらに、上記デコード命令列を格納するための実マシンコード領域８６０７ａ～８６０７ｃと、当該実マシンコード領域８６０７ａ～８６０７ｃにデコード命令列が格納されているか否かを示すフラグを格納するフラグ領域８６０５ａ～８６０５ｃとから構成される。例えば、図５３（ｂ）に示された命令ブロック格納領域３９５２ｂは、実マシンコード領域８６０７ｂにデコード命令列が存在しないので、その旨（空）のフラグがフラグ領域８６０５ｂに格納され、一方、図５３（ｃ）に示された命令ブロック格納領域３９５２ｃは、実マシンコード領域８６０７ｃにデコード

命令列が存在するので、その旨（あり）のフラグがフラグ領域８６０５ｃに格納されている。

【０１７２】

なお、各実マシンコード領域に格納すべきデコード命令列は、例えば、実施形態５に係る仮想マシン３８００を用いることによって予め得ることができる。このデコード命令列は、各命令ブロックの仮想マシンプログラムを実施形態５に係る仮想マシン３８００が実行した場合に、そのデコード部３８０２が実行部３８１０に出力するデコードデータ列に等しいからである。

【０１７３】

また、各命令ブロックにおいて、仮想マシンコード領域３９５４ａ～３９５４ｄ、３９５５ａ～３９５５ｄに置かれている個々の仮想マシン命令と、実マシンコード領域８６０７ａ～８６０７ｄに置かれている対応するデコードデータとは、一定のオフセットだけ離れたアドレスの位置に配置されている。

カレントフラグ記憶部３９０７は、本仮想マシン３９００の実行対象となっている命令格納部３９０１中の命令ブロックのフラグ領域に格納されているフラグをカレントフラグとして保持するための一時的な記憶領域である。

【０１７４】

命令読込み部３９０３は、カレントフラグ記憶部３９０７に記憶されたフラグの値に基づいて、命令格納部３９０１から仮想マシン命令又はデコードデータを読み出し、それぞれ、検索部４４０５又は実行部３９１０に出力する。つまり、デコードデータを読み込んだ場合には、検索部４４０５をバイパスして直接実行部３９１０にデコードデータを渡す。

【０１７５】

カレントフラグ読出制御部３９１２は、デコード部３９０２から送られてくるデコードデータが分岐命令に相当するか監視し、分岐命令の場合には、その分岐命令の実行直後に、デコード部３９０２を制御することにより、その分岐先の命令ブロックのフラグ領域に格納されたフラグを読み出してカレントフラグ記憶部３９０７に格納させる。

（仮想マシンの動作）

以上のように構成された仮想マシン３９００の動作について説明する。

【０１７６】

図５４は、デコード部３９０２の動作を示すフローチャートである。

デコード部３９０２の命令読込み部３９０３は、信号線Ｒを介して実行部３９１０から次の仮想マシン命令を読み込む旨の通知を受けると（ステップ８７０２、８７０３）、カレントフラグ記憶部３９０７に記憶されたフラグを読み出して、その内容を判断する（ステップ８７０４）。

【０１７７】

その結果、デコード命令列が存在しないと判断した場合には、命令読込み部３９０３は、実施形態５の場合と同様の動作、即ち、ＰＣ３８０４に格納された仮想マシンコード領域上のアドレスに従って非分岐命令格納領域又は分岐命令格納領域に格納された仮想マシンコードを読み出し、検索部４４０５に渡す（ステップ８７０５、８７０６）。そして、検索部４４０５は、デコードテーブル４４０６を検索することでジャンプアドレスを特定し、デコードデータとして実行部３９１０に出力した後に（ステップ８７０７）、その旨を信号線Ｒで通知する（ステップ８７１１）。

【０１７８】

一方、上記カレントフラグによってデコード命令列が存在すると判断した場合には、命令読込み部３９０３は、ＰＣ３８０４に格納された仮想マシンコード領域上のアドレスに一定のオフセットを加算することで実マシンコード領域８６０７ａ～８６０７ｄ上のアドレスを算出した後に（ステップ８７０８）、そのアドレスに従ってデコードデータを読み出し（ステップ８７０９）、それを直接実行部３９１０に出力する（ステップ８７１０）。

【０１７９】

図５５は、実行部３９１０の動作を示すフローチャートである。

本図と図７９と比較して分かるように、基本的な流れは従来と同じである。つまり、ＰＣ３８０４やＳＰ４４１２を初期化した後に（ステップ８８０２）、デコード部３９０２から送られてくるデコードデータに基づいてマイクロプログラム記憶部４４１１内のマイクロプログラムを実行する（ステップ８８０４～８８

０８）。

【０１８０】

異なる点は、カレントフラグ記憶部３９０７についての処理（ステップ８８０３）が追加されていることである。つまり、実行部３９１０は、動作の開始にあたって、デコード命令列が存在しない旨のフラグを初期値としてカレントフラグ記憶部３９０７に格納しておく（ステップ８８０３）。

図５６は、実行部３９１０が分岐命令を実行した場合のデコード部３９０２に対する制御を示すフローチャートである。本図と図４２とを比較して分かるように、実行部３９１０が分岐命令を実行する場合においては、分岐先変換部３８１１は、分岐命令のオペランドを分岐先と命令ブロックの識別子セグメントとオフセットに変換し、それぞれＰＣ３８０４の識別子セグメントレジスタ３８０４ａとオフセットカウンタ３８０４ｂに格納するが（ステップ８９０２、８９０４）、この処理は実施形態５の場合と同じである。

【０１８１】

異なる点は、カレントフラグ記憶部３９０７についての処理（ステップ８９０４）が追加されていることである。つまり、分岐先変換部３８１１によってＰＣ３８０４が更新された後に（ステップ８９０２、８９０４）、カレントフラグ読出制御部３９１２は、命令読込み部３９０３を制御することによって、識別子セグメントレジスタ３８０４ａに格納された識別子セグメントが示す命令ブロックのフラグ領域の値を読み出し、カレントフラグ記憶部３９０７に格納させる（ステップ８９０４）。これによって、新たな命令ブロックに分岐する度に、カレントフラグ記憶部３９０７の内容は更新され、これから実行する命令ブロックの実マシンコード領域にデコード命令列が格納されているか否かを示すフラグがカレントフラグ記憶部３９０７にセットされる。

【０１８２】

以上のように、本実施形態に係る仮想マシン３９００によれば、命令格納部３９０１には、実行対象となる仮想マシンプログラムが基本ブロックを単位とする命令ブロックに分割されて格納されており、さらに、各命令ブロックには仮想マシン命令だけでなく、それら仮想マシン命令に対応するデコードデータも配置さ

れる。そして、デコード部３９０２は、各命令ブロックのフラグ領域を参照することで、デコードデータが置かれた命令ブロックについては、単にそれらデコードデータを読み出して実行部３９１０に渡すだけで足り、検索部４４０５によるデコードテーブル４４０６の検索処理は省略される。これにより、実施形態５に係る仮想マシン３８００と同等の効果が得られることに加え、予めデコード命令列が配置された命令ブロックについての実行時間は短縮される。

【０１８３】

なお、本実施形態では、各命令ブロックにおける仮想マシンコード領域と実マシンコード領域とは、そのアドレスが一定のオフセットだけ離れているという位置関係にあったが、例えば、デコード命令列格納領域の先頭位置を特定するオフセットアドレスを各命令ブロック中に配置しておくことで、これら位置関係による制約を不要とすることができる。その場合には、新たな命令ブロックに分岐する度に、その命令ブロックのフラグと共に、そのオフセットアドレスを読み出すことで、カレントフラグに応じた仮想マシンコード領域及び実マシンコード領域それぞれに適したアドレスをＰＣ３８０４にセットすることができる。

（実施形態７）

次に、上記実施形態６に係る仮想マシンにおけるデコード命令列を動的に生成する実施形態７に係る仮想マシンについて説明する。

（仮想マシンの構成）

図５７は、本実施形態に係る仮想マシン４０００の構成を示す機能ブロック図である。本仮想マシン４０００は、命令格納部３９０１、デコード部４００２、実行部３９１０及びスタック４４２０から構成される。

【０１８４】

本図と図５２とを比較して分かるように、本仮想マシン４０００は、実施形態６に係る仮想マシン３９００とほぼ同一の構成要素を備える。相違する点は、デコード部４００２にデコード命令列書込み部４００８が追加されていること、その追加に伴うデコード部４００２内部の接続変更である。以下、実施形態６に係る仮想マシン３９００と相違する点を中心に説明する。

【０１８５】

デコード命令列書込み部４００８は、本仮想マシン４０００による実行制御がデコード命令列を持たない命令ブロックに分岐した場合に、その命令ブロックに対する実行を中断させ、一旦、その命令ブロックに置かれている仮想マシンプログラム全体を命令読込み部３９０３及び検索部４４０５によってデコード命令列に変換させた後に、そのデコード命令列を当該命令ブロックのデコード命令列格納領域に書き込む。そして、書き込まれたデコード命令列に対して、命令読込み部３９０３による読込みと実行部３９１０による実行を再開させる。

【０１８６】

従って、結果的に、命令格納部３９０１から命令読込み部３９０３によって読み出されたデコードデータだけがそのまま実行部３９１０に渡され、検索部４４０５がデコードテーブル４４０６を検索することによって得たデコードデータは直接には実行部３９１０に渡されない。このことは、実施形態６の場合と相違し、検索部４４０５からのデコードデータは、実行部３９１０ではなく、デコード命令列書込み部４００８に送られていることに対応する。

（仮想マシンの動作）

以上のように構成された仮想マシン４０００の動作について説明する。

【０１８７】

図５８は、本仮想マシン４０００が分岐命令を実行した場合の本仮想マシン４０００における特徴的な動作、即ち、デコード命令列書込み部４００８、カレントフラグ読出制御部３９１２及び分岐先変換部３８１１における動作を示すフローチャートである。新たな命令ブロックへの分岐に際して、分岐先変換部３８１１がＰＣ３８０４の値を更新し（ステップ９１０２、９１０３）、カレントフラグ読出制御部３９１２がカレントフラグ記憶部３９０７の内容を更新する点は、図５６に示された実施形態６の手順と同じである。異なる点は、その後に、必要に応じて、デコード命令列書込み部４００８によるデコード命令列の生成と命令格納部３９０１への書き込みが行われることである（ステップ９１０５～９１１１）。

【０１８８】

具体的には、デコード命令列書込み部４００８は、命令読込み部３９０３によ

って読み出されたフラグを受け取って参照することによって、その命令ブロックには既にデコード命令列が格納されているか否か判断する（ステップ９１０５）。

その結果、デコード命令列が格納されていると判断した場合には、デコード命令列書込み部４００８は、特になんの処理も行わない（ステップ９１１２）。このときには、その命令ブロックに格納されているデコード命令列が順次読み出されて実行部３９１０によって直接に実行されることになる。

【０１８９】

一方、デコード命令列が格納されていないと判断した場合には、デコード命令列書込み部４００８は、ポインタdPCをインクリメントしながら（ステップ９１０６～９１１１）、その命令ブロックに置かれている仮想マシンコードを命令読込み部３９０３によって順次読み出させ（ステップ９１０８、９１０９）、読み出された仮想マシンコードを検索部４４０５によって必要なジャンプアドレス等のデコードデータに変換させた後に当該命令ブロックのデコード命令列格納領域に書き込んでいく（ステップ９１１０）。

【０１９０】

そして、当該命令ブロックの全ての仮想マシンコードについてデコードデータへの変換と書き込みを終えると（ステップ９１０７）、デコード命令列が存在する旨のフラグをカレントフラグ記憶部３９０７及び当該命令ブロックのフラグ領域に書き込むことで、処理を終える（ステップ９１１２）。これによって、当該命令ブロックのデコード命令列に対する命令読込み部３９０３による読込みと実行部３９１０による実行が再開される。

【０１９１】

図５９は、図５８におけるステップ９１１０、即ち、仮想マシンコードからデコードデータへの変換と命令格納部３９０１への格納処理の詳細を示すフローチャートである。本図と図７７とを比較して分かるように、従来の検索部４４０５による処理に、デコード命令列書込み部４００８による処理、つまり、デコードテーブル４４０６の検索によって得られたジャンプアドレスddや仮想マシン命令のオペランドをデコードデータとして命令格納部３９０１のデコード命令列格納

領域に書き込む処理（ステップ９２０９、９２１３）が追加されていることがわかる。

【０１９２】

図６０は、実行部３９１０から見たデコード部４００２の動作を示すフローチャートである。命令読込み部３９０３は、実行部３９１０との関係においては、命令格納部３９０１の実マシンコード領域からデコードデータを読み出した場合にだけ、そのデコードデータを実行部３９１０に渡すので、デコードデータ専用の読み出し部として機能していると言える。

【０１９３】

以上のように、本実施形態に係る仮想マシン４０００によれば、デコード命令列を持たない命令ブロックに分岐した場合に、一旦、その命令ブロックの仮想マシンプログラムはデコードデータに変換され、命令格納部３９０１に書き戻された後に、そのデコードデータが直接に実行される。従って、再び、その命令ブロックを実行する場合には、デコードデータを読み出して直接に実行することができるので、解読のための時間、即ち、検索部４４０５によるデコードテーブル４４０６の検索に伴う時間が削減される。特に、ループ処理のように、同一の命令ブロックを繰り返して実行する場合における実行速度は飛躍的に向上される。

（実施形態８）

次に、上記実施形態７に係る仮想マシンを圧縮符号化された仮想マシンプログラムに対応させた実施形態８に係る仮想マシンについて説明する。

（仮想マシンの構成）

図６１は、本実施形態に係る仮想マシン４１００の構成を示す機能ブロック図である。本仮想マシン４１００は、命令格納部４１０１、デコード部４１０２、実行部３９１０及びスタック４４２０から構成される。

【０１９４】

本図と図５７とを比較して分かるように、本仮想マシン４１００は、実施形態７に係る仮想マシン４０００とほぼ同一の構成要素を備える。相違する点は、命令格納部４１０１に格納されている仮想マシンプログラムのコード形態、命令格納部４１０１に展開情報格納領域４１５７ａ～４１５７ｄが設けられていること

、デコード部４１０２の命令読込み部４１０３に仮想マシン命令展開部４１０３ａが追加されていることである。以下、実施形態７に係る仮想マシン４０００と相違する点を中心に説明する。

【０１９５】

命令格納部４１０１の非分岐命令格納領域４１５４ａ～４１５４ｄ及び分岐命令格納領域４１５５ａ～４１５５ｄ（これら領域を併せて、本実施形態では、「圧縮仮想マシンコード領域」と呼ぶ。）は、圧縮符号化された仮想マシン命令を予め保持している。また、命令格納部４１０１の展開情報格納領域４１５７ａ～４１５７ｄは、その命令ブロックに格納された圧縮符号化された仮想マシン命令を伸長復号化するための復号化テーブルを予め保持している。

【０１９６】

図６２（ａ）は、その復号化テーブルの例を示す。圧縮されたビット列と、対応する仮想マシン命令との組が示されている。

図６２（ｂ）は、図６２（ａ）に示された復号化テーブルにおける符号の規則性を示す図である。ここでは、ハフマン符号化に準じたビット圧縮方法で、オペランドを含む仮想マシン命令の単位でビット列に圧縮していくものとする。例えば、ビット列"000"は仮想マシン命令"Push [0]"を意味し、ビット列"01010"は仮想マシン命令"Push 10"を意味する。

【０１９７】

図６３（ａ）～（ｃ）は、命令格納部４１０１に格納されている仮想マシンプログラムの格納状態の例を示す図であり、図９７に示されたサンプル仮想マシンプログラムが格納されている場合に相当する。各命令ブロック格納領域４１５２ａ～４１５２ｃにおいて、非分岐命令格納領域４１５４ａ～４１５４ｃ及び分岐命令格納領域４１５５ａ～４１５５ｃからなる各圧縮仮想マシンコード領域４１６４ａ～４１６４ｃには、各命令ブロックの仮想マシンプログラムを圧縮符号化して直列に接続することによって得られるビット列（以下、「圧縮ビット列」と呼ぶ。）が予め格納され、各展開情報格納領域４１５７ａには、当該ビット列を伸長復号化するための復号化テーブルが格納されている。なお、図６３（ｂ）には、デコード命令列を持たない命令ブロック格納領域４１５２ｂが示され、図６

３（ｃ）には、デコード命令列を持つ命令ブロック格納領域４１５２ｃが示されている。　命令読込み部４１０３は、実施形態７の命令読込み部３９０３が備える機能、即ち、命令格納部４１０１の圧縮仮想マシンコード領域４１６４ａ～４１６４ｄから圧縮ビット列を読み出し、デコード命令列格納領域４１５６ａ～４１５６ｄからデコード命令列を読み出すことに加え、さらに、仮想マシン命令展開部４１０３ａを備える。

【０１９８】

仮想マシン命令展開部４１０３ａは、命令読込み部４１０３が命令格納部４１０１の圧縮仮想マシンコード領域４１６４ａから圧縮ビット列を１ビットずつ読み出す度にその命令ブロックの展開情報格納領域４１５７ａ～４１５７ｄに格納された復号化テーブルを参照することで、対応する仮想マシン命令を特定し、特定した仮想マシン命令を検索部４４０５に渡すという伸長復号化を繰り返す。

（仮想マシンの動作）

以上のように構成された仮想マシン４１００の動作について説明する。

【０１９９】

上述したように、本仮想マシン４１００は、実施形態７に係る仮想マシン４０００が備える機能を全て含むので、圧縮ビット列の復号処理を除く全体的な処理は実施形態７と同じであり、図５８に示されるフローチャートの通りである。

つまり、本仮想マシン４１００は、実施形態７に係る仮想マシン４０００と同様に、デコード命令列を持たない命令ブロックに分岐した場合には、命令読込み部４１０３及び検索部４４０５によって、一旦、その命令ブロックの仮想マシンプログラムをデコードデータに変換し、デコード命令列書込み部４００８によって命令格納部３９０１に書き戻した後に、命令読込み部４１０３及び実行部３９１０によってデコード命令列を直接に実行する。

【０２００】

ところが、本実施形態の仮想マシン４１００は、圧縮符号化された仮想マシン命令を読み出すので、図５８におけるステップ９１０９とステップ９１１０での詳細な処理が実施形態７と異なる。圧縮ビット列に適した読み出しを行う必要があること、及び、その復号化処理を追加する必要があるためである。

図６４は、図５８におけるステップ９１０９及びステップ９１１０の詳細を示すフローチャートであり、本仮想マシン４１００のデコード部４１０２の動作を示す。ここで、図６４におけるステップ９６０２、ステップ９６０３～９６１６は、それぞれ、図５８におけるステップ９１０９、ステップ９１１０に相当する。

【０２０１】

本図と実施形態７における図５９とを比較して分かるように、相違する点は、本実施形態では、直接に仮想マシンコードを読み出すことに替えて、圧縮ビット列の読み出しと復号化を行っていることと（ステップ９６０２）、その復号化において必要なオペランド（配列op[i]）が同時に得られるので、命令格納部４１０１からオペランドだけを読み出すことなくそのオペランド（配列op[i]）をデコード命令列格納領域４１５６ａ～４１５６ｄに書き込んでいること（ステップ９６１３）である。

【０２０２】

図６５は、図６４におけるステップ９６０２の詳細を示すフローチャートである。命令読込み部４１０３は、まず、圧縮ビット列の一時的な記憶域（変数bits）を確保した後に（ステップ９７０２）、ＰＣ３８０４の値に従って、デコード命令列を持たない命令ブロック格納領域４１５２ａ～４１５２ｄの圧縮仮想マシンコード領域４１６４ａ～４１６４ｄから１ビット分の圧縮符号を読み出し（ステップ９８０３）、それと既に読み出した圧縮符号（変数bits）とを連結する（ステップ９７０４）。

【０２０３】

そして、仮想マシン命令展開部４１０３ａは、ＰＣ３８０４の値に一定のオフセットを加算することによって得たアドレスから始まる展開情報格納領域４１５７ａ～４１５７ｄの復号化テーブル、即ち、読み出された圧縮ビット列が置かれていた命令ブロックの復号化テーブルに登録されている圧縮ビット列それぞれと上記ステップ９７０４で得られた圧縮符号（変数bits）とを順次に比較することで、一致する仮想マシン命令を特定する（ステップ９７０５）。これら読み出し（ステップ９７０３）と検索（ステップ９７０５）は、一致する仮想マシン命令

が見つかるまで繰り返す（ステップ９７０６）。

【０２０４】

一致する仮想マシン命令が見つかると、仮想マシン命令展開部４１０３ａは、その仮想マシン命令を展開情報格納領域４１５７ａ～４１５７ｄから読み出し（ステップ９７０７）、その仮想マシン命令を、又は、その仮想マシン命令にオペランドが含まれる場合にはその仮想マシン命令中のオペコードとオペランド（配列op[]）とを分離した後に検索部４４０５に出力する（ステップ９７０８、９７０９）。その後、図６４のステップ９６０３～９６１４に示されるように、検索部４４０５によって対応するデコードデータに変換され、デコード命令列書込み部４００８によって、必要なオペランド配列op[]と共に当該命令ブロックの実マシンコード領域に書き込まれる。

【０２０５】

以上のように、本実施形態に係る仮想マシン４１００によれば、命令格納部４１０１の各命令ブロックには圧縮
符号化された仮想マシンプログラムが置かれ、本仮想マシン４１００がデコード命令列を持たない命令ブロックに分岐した場合には、一旦、その命令ブロックに置かれた圧縮符号化された仮想マシンプログラムが伸長復号化された後に、デコードデータに変換されて命令格納部３９０１に書き戻され、そのデコードデータが直接に実行される。

【０２０６】

従って、本仮想マシン４１００によれば、圧縮ビット列は、常に命令ブロックの先頭、即ち、完全な１個の仮想マシン命令の先頭から復号されることが保証される。これによって、分岐命令の実行に伴って１個の仮想マシン命令の圧縮ビット列の途中から誤って復号化してしまうことが確実に回避され、圧縮符号化された仮想マシンプログラムを正しく実行することが可能な仮想マシンが実現される。

【０２０７】

なお、本実施形態では、命令格納部４１０１の各命令ブロック格納領域４１５２ａ～４１５２ｄにはデコード命令列格納領域４１５６ａ～４１５６ｄが設けら

れていたが、圧縮ビット列の途中から誤って復号化されるとい従来の問題点を回避するだけならば、これらデコード命令列格納領域４１５６ａ～４１５６ｄを設けなくてもよい。

【０２０８】

つまり、本実施形態に係る仮想マシン４１００は、デコード命令列格納領域４１５６ａ～４１５６ｄを備えた実施形態７に係る仮想マシン４０００を、圧縮符号化された仮想マシンプログラムを解読実行するように対応させたものに相当するが、デコード命令列格納領域を備えない実施形態５に係る仮想マシン３８００を、圧縮符号化された仮想マシンプログラムを解読実行するように対応させものであってもよい。いずれの場合であっても、圧縮符号化された仮想マシンプログラムは基本ブロックを単位として格納され、全ての分岐命令が指定する分岐先は基本ブロックの先頭であることが保証されているので、圧縮ビット列の途中から誤って復号されてしまうことは回避されるからである。

【０２０９】

なお、本実施形態では、圧縮符号化の手法としてハフマン符号化を用いたが、LZ法等の他の圧縮手法を用いても、なんら問題がないのは言うまでもない。

（実施形態９）

次に、対象とする実マシン固有のジャンプ先についての境界制限を満足する実マシンコードを高速に生成する実施形態９に係るその場式コンパイラについて説明する。

（コンパイラシステムの構成）

図６６は、本実施形態に係るその場式コンパイラ４３００を含むコンパイラシステム全体の構成を示す機能ブロック図である。つまり、本図には、その場式コンパイラ４３００だけでなく、その場式コンパイラ４３００への入力に必要な情報を生成する仮想マシンコンパイラ４３２０も併せて示されている。

【０２１０】

仮想マシンコンパイラ４３２０は、Ｃ言語等の高級言語で記述されたソースプログラム４３１０を入力とし、特定の仮想マシンを対象とする仮想マシンコードを生成して経路D1に出力する一般的なコンパイラが備える言語変換機能に加えて

、その場式コンパイラ４３００で必要とされる特有の情報（ブロック先頭情報）を生成し経路D2に出力するブロック先頭情報生成部４３２１ａをも備える。

【０２１１】

ブロック先頭情報生成部４３２１ａは、一般的なコンパイラが備える出力部４３２１、つまり、構文解析や中間コードへの変換等を経て得られた最終的な仮想マシンコードを外部に順次出力する出力部４３２１に付加された機能であり、出力部４３２１から経路D1に出力される各仮想マシンコードごとに、その仮想マシンコードが基本ブロックの先頭とすべきものか否か判断し、その結果を示すブロック先頭情報を経路D2に出力する。

【０２１２】

その場式コンパイラ４３００は、上記仮想マシンコンパイラ４３２０が生成した仮想マシンコード及びブロック先頭情報を入力とし、実マシン命令の分岐先がアドレス空間の２ワード境界に制限された実マシンを対象とする実マシン命令列４３１１に変換するコンパイラであり、実マシン命令変換部４３０１、分岐位置補正部４３０２及び実マシンアドレス保持部４３０３とから構成される。

【０２１３】

実マシン命令変換部４３０１は、仮想マシンコンパイラ４３２０から経路D1を経て出力された仮想マシンコードがオペコードの場合には内部の変換テーブルに基づいて対応する実マシンコードに変換し、一方、その仮想マシンコードがオペランドの場合にはそのまま、分岐位置補正部４３０２に出力する。そのとき、実マシン命令変換部４３０１は、実マシンアドレス保持部４３０３が保持する実マシンアドレスPCを読み出して、上記実マシンコードと共に分岐位置補正部４３０２に出力した後に、その実マシンアドレスPCをインクリメントしておく。

【０２１４】

実マシンアドレス保持部４３０３は、実マシン命令変換部４３０１が次に生成する実マシンコードを配置すべき実マシン空間での相対アドレスPCを保持する。

分岐位置補正部４３０２は、実マシン命令変換部４３０１から送られてきた実マシンアドレスPCと仮想マシンコンパイラ４３２０から経路D2を経て出力されたブロック先頭情報とから、基本ブロックの先頭に位置する実マシン命令が奇数番

地、即ち、２ワード境界を違反する位置に配置されることとなるか否かを判断し、違反することとなる場合には、１ワード分のダミー命令、即ち、無動作の実マシン命令"Nop"を前置させた後に、実マシン命令変換部４３０１から送られてきた実マシンコードを実マシン命令列４３１１として外部に出力する。これは、処理内容を変更することなく、基本ブロックの実質的な先頭を２ワード境界に配置させるためである。

（コンパイラシステムの動作）

以上のように構成されたコンパイラシステムの動作について、一般的なコンパイラと異なる点を中心に説明する。

【０２１５】

図６７は、仮想マシンコンパイラ４３２０のブロック先頭情報生成部４３２１ａの動作を示すフローチャートである。図４７に示された実施形態５に係る仮想マシンコンパイラの動作と基本的な流れは同じである。

まず、ブロック先頭情報生成部４３２１ａは、出力部４３２１が出力しようとする仮想マシンコードVCそれぞれについて、基本ブロックの先頭とすべきものかどうかを判定する（ステップ１０００３、１０００４）。その結果、基本ブロックの先頭とすべき仮想マシンコードと判定した場合にはその旨を示すブロック先頭情報"T"を（ステップ１０００６）、一方、先頭とすべきでないと判定した場合にはその旨を示すブロック先頭情報"N"を（ステップ１０００５）、その仮想マシンコードVCと共に、それぞれ、経路D1及びD2に出力する（ステップ１０００７）。

【０２１６】

図６８は、実マシン命令変換部４３０１、分岐位置補正部４３０２及び実マシンアドレス保持部４３０３の動作を示すフローチャートである。まず、実マシンアドレス保持部４３０３は、実マシンアドレスPCを初期化しておく（ステップ１０１０２）。

実マシン命令変換部４３０１は、ブロック先頭情報生成部４３２１ａから出力された仮想マシンコードVCを受け取ると（ステップ１０１０３、１０１０４）、必要に応じて対応する実マシンコードに変換し、それを実マシンアドレス保持部

４３０３から読み出した実マシンアドレスPCと共に分岐位置補正部４３０２に渡した後に、実マシンアドレスPCをインクリメントしておく（ステップ１０１０５）。

【０２１７】

次に、分岐位置補正部４３０２は、上記仮想マシンコードVCに対応するブロック先頭情報BIをブロック先頭情報生成部４３２１ａから受け取ると（ステップ１０１０６）、実マシン命令変換部４３０１から送られてきた実マシンコードを外部に出力するに際して、上記境界違反を生じることとなるか否か判断する（ステップ１０１７、１０１０８）。具体的には、ブロック先頭情報生成部４３２１ａから受け取ったブロック先頭情報BIが基本ブロックの先頭である旨"T"を示し、かつ、実マシン命令変換部４３０１から受け取った実マシンアドレスPCが２ワード境界にないか否かを判断する（ステップ１０１０７、１０１０８）。

【０２１８】

その結果、基本ブロックの先頭であり、かつ、２ワード境界にないと判断した場合には、上記境界違反を回避するために、実マシン命令"Nop"を生成し出力した後に上記実マシン命令を実マシン命令列４３１１として出力し（ステップ１０１０９、１０１１０）、そうでない場合には、単に上記実マシン命令を実マシン命令列４３１１として出力する（ステップ１０１１０）。なお、分岐位置補正部４３０２は、実マシン命令"Nop"を生成した場合には（ステップ１０１１０）、そのサイズ分だけ実マシンアドレス保持部４３０３の実マシンアドレスPCを更新しておく。

【０２１９】

以上の処理（ステップ１０１０４～１０１１０）は、ブロック先頭情報生成部４３２１ａから仮想マシンコードが送られてくる限り、繰り返す（ステップ１０１０３、１０１１１）。

図６９は、図９７に示されたサンプル仮想マシン命令列が本その場式コンパイラ４３００に入力された場合において、ブロック先頭情報生成部４３２１ａによって生成されるブロック先頭情報、分岐位置補正部４３０２によって生成される実マシン命令"Nop"のタイミング、その他の関連情報を示すテーブルである。本

図から分かるように、仮想マシンアドレス０番、アドレス８番、アドレス１５番及びアドレス３１番の仮想マシン命令は、基本ブロックの先頭となるので、その旨のブロック先頭情報"T"が生成される。

【０２２０】

また、仮想マシンアドレス１５番においては、分岐位置補正部４３０２は、ブロック先頭情報生成部４３２１ａからブロック先頭情報"T"を受け取り、実マシン命令変換部４３０１から奇数（３５）の実マシンアドレスPCを受け取るので、仮想マシン命令"Push [1]"に対応する実マシン命令を出力する前に、実マシン命令"Nop"を出力する。これによって、ブロックの先頭命令が奇数番地に置かれることが回避される。

【０２２１】

以上のように、本実施形態に係るその場式コンパイラ４３００は、従来において必要とされた分岐命令の分岐先についての解析という複雑な処理を行うことなく、ジャンプ先についての境界違反を起こすことがない実マシンプログラムを生成することができた。これは、仮想マシンコンパイラ４３２０において、ブロック先頭情報生成部４３２１ａが基本ブロックの検出を行い、その検出結果であるブロック先頭情報をその場式コンパイラ４３００に伝達しているからである。

【０２２２】

従って、本発明に係るその場式コンパイラ４３００は、従来のその場式コンパイラと比較し、ブロック先頭情報に基づく実マシン命令"Nop"の追加挿入という簡単な処理を追加するだけで、境界違反の問題を解消している。これによって、ジャンプ命令のジャンプ先についての境界制限を違反することがない適正な実マシンコードが短時間で生成されるその場式コンパイラが実現される。

【０２２３】

なお、本実施形態に係るブロック先頭情報生成部４３２１ａは、仮想マシンコンパイラ４３２０の出力部４３２１に付随して備えられたが、これに代えて、一般的なコンパイラが備える基本ブロックへの分割処理を利用してもよい。つまり、一般的なコンパイラは、通常、最適化等の過程において、基本ブロックへの分割処理を行うので、その分割処理の過程で得られたブロック先頭情報を外部（そ

の場式コンパイラ４３００）に出力させることで、容易にブロック先頭情報生成部４３２１ａを実現することができる。

【０２２４】

また、本実施形態では、ジャンプ先となる実マシン命令の配置についてのみアライメント処理が行われたが、同様な方法によって、遅延分岐や打ち消し分岐のための配置を行うことができるのは言うまでもない。

以上、本発明に係る仮想マシン、仮想マシンコンパイラ及びその場式コンパイラについて、実施形態１～９に基づいて説明したが、本発明は、これら実施形態に限定されるものではない。各実施形態に示された特徴的な構成要素を組み合わせたり分離したりすることで、容易に本発明のバリエーションを考案することができる。

【０２２５】

例えば、実施形態１と実施形態５とを組み合わせることで、仮想マシンプログラムが基本ブロック単位で分割され、かつ、対応する先行命令情報と共に命令格納部に格納しておくことができる。これによって、真のデータ依存関係を解消し、かつ、その場式コンパイラによるアドレス解決処理を単純化する高速な仮想マシンが実現される。

【０２２６】

また、実施形態２と実施形態８とを組み合わせることで、適度な頻度で割込み処理が行われ、かつ、適正に復号されることが保証された圧縮ビット列を実行する割込み機能付き仮想マシンが実現される。

また、実施形態１における先行命令情報と仮想マシン命令、実施形態９におけるブロック先頭情報と仮想マシン命令とは、それぞれ独立分離された構造であったが、例えば、図７０に示されるように、本発明に係る仮想マシンが実行する仮想マシン命令として、これら先行命令情報、ブロック先頭情報が埋め込まれた命令フォーマットを有する拡張仮想マシン命令を定義してもよい。その場合には、拡張仮想マシン命令を単位として命令格納部等から読み込んだ後に、定型的に分割することで、先行命令情報、ブロック先頭情報、仮想マシン命令のオペコード及びオペランドを区別して得ることができる。

【０２２７】

また、実施形態５～８では、各命令ブロック格納部はユニークな識別子を保持していたが、各命令ブロックを一定のルールに従って命令格納部に配置する等により、それらを個別に識別できるならば、各命令ブロック格納部に識別子を保持させる必要はない。

さらに、本発明に係る仮想マシン、仮想マシンコンパイラ及びその場式コンパイラは、いずれも、汎用のコンピュータによって実行されるプログラムとして実現することができるので、これら発明に係るプログラムをCD-ROM等の記録媒体に収納したり、伝送路を介して通信したりすることによって、流通させることができるのは言うまでもない。

【０２２８】

【発明の効果】

以上の説明から明らかなように、本発明に係る仮想マシンは、実マシンによる制御の下で仮想マシン命令を実行するスタック型仮想マシンであって、データを後入れ先出し方式で一時的に記憶するためのスタック手段と、実行対象となる仮想マシン命令列と、それら仮想マシン命令それぞれに対応づけられた情報であって、対応づけられた仮想マシン命令に後続する仮想マシン命令が実行された場合の前記スタック手段でのデータの格納状態の変化を示す先行命令情報とを記憶する命令記憶手段と、前記命令記憶手段から次に実行すべき仮想マシン命令及び対応する先行命令情報を読み出す読み出し手段と、読み出された仮想マシン命令と先行命令情報との組み合わせに対応する演算処理を特定し実行する解読実行手段とを備えることを特徴とする。

【０２２９】

これによって、命令記憶手段には、仮想マシン命令だけでなく先行命令情報をも格納されているので、解読実行部は、仮想マシン命令の演算処理と共に、後続する仮想マシン命令のためのスタック操作を先行して処理しておくことができるので、例えば、スーパースカラマシンにおいて特に生じ易いパイプラインハザードのマシンサイクルにおいてその先行処理を実行しておくことにより、真のデータ依存関係の影響が吸収され、実行速度が高速化されるという効果がある。

【０２３０】

ここで、前記解読実行手段は、対象とする全ての種類の仮想マシン命令と全ての種類の先行命令情報との組み合わせそれぞれに対応する実マシン命令列を記憶する実マシン命令列記憶部と、前記読み出し手段によって読み出された仮想マシン命令と先行命令情報との組み合わせに対応する実マシン命令列を前記実マシン命令列記憶部に記憶された実マシン命令列から特定する特定部と、特定された実マシン命令列を実行する実行部とを有することとすることもできる。

【０２３１】

これによって、データ依存関係を吸収するための上記先行処理を各仮想マシン命令に対応する実マシン命令列に組み入れておくことができる。

また、前記先行命令情報は、対応する仮想マシン命令に後続する仮想マシン命令が実行された場合の前記スタック手段に格納されるデータ数の増減を示し、前記実マシン命令列記憶部に記憶された実マシン命令列には、対応する先行命令情報に基づいて前記スタック手段を先行処理する実マシン命令が含まれていることとすることもできる。

【０２３２】

これによって、当該仮想マシン命令の実行に伴うスタック段数の増減と、後続する仮想マシン命令の実行に伴うスタック段数の増減とが、それぞれ打ち消し合う操作である場合においては、無駄なスタック操作が回避され、実行速度が向上される。

また、前記実マシン命令列記憶部に記憶された実マシン命令列は、前記スタック手段において最後に格納されたデータと最後から２番目に格納されたデータそれぞれを記憶する領域が前記実マシンが備える２つのレジスタに割り当てられていることを前提とする内容であるとすることもできる。

【０２３３】

これによって、スタック型仮想マシンにおいて頻繁に行われるスタックへのデータの出し入れは、実マシンの内部レジスタへの読み書き動作となるので、パイプラインハザードを埋めるのに好適な先行処理となり、実行効率の向上に適したアーキテクチャとなる。

また、前記命令記憶手段は、前記仮想マシン命令列を記憶する第１領域と、その第１領域における各仮想マシン命令の記憶位置に関連付けられた記憶位置に前記先行命令情報を記憶する第２領域とからなり、前記読み出し手段は、前記第１領域と前記第２領域それぞれの対応する記憶領域から前記仮想マシン命令と先行命令情報との組を読み出すこととすることもできる。

【０２３４】

これによって、仮想マシン命令列は、先行命令情報と分離されて格納されるので、従来の仮想マシン命令列と同一のフォーマットとあんり、従来の仮想マシンとの命令フォーマットの互換性が維持される。

また、前記命令記憶手段は、前記仮想マシン命令列と前記先行命令情報とを、対応する１つの仮想マシン命令と１つの先行命令情報との組を１つの拡張仮想マシン命令とする拡張仮想マシン命令列として記憶し、前記読み出し手段は、前記命令記憶手段から拡張仮想マシン命令を読み出し、前記解読実行手段は、読み出された拡張仮想マシン命令に対応する演算処理を特定し実行することとすることもできる。

【０２３５】

これによって、拡張仮想マシン命令を新たな１個の仮想マシン命令と扱うことで、先行命令情報だけを特別に処理したり記憶させたりする処理が不要となり、全体として、従来のコンピュータのアーキテクチャに近い仮想マシンが実現される。

また、本発明に係るコンパイラは、スタック型仮想マシンを対象とするコンパイラであって、ソースプログラムを前記仮想マシンが実行する仮想マシン命令列に変換する命令列変換手段と、変換によって得られた仮想マシン命令列を構成する仮想マシン命令それぞれについて、後続する仮想マシン命令が前記仮想マシンによって実行された場合の前記スタック手段でのデータの格納状態の変化を示す先行命令情報を生成する先行命令情報生成手段と、生成された先行命令情報と対応する仮想マシン命令とを関連付けて出力する関連付け手段とを備えることを特徴とする。

【０２３６】

これによって、仮想マシン命令だけでなく先行命令情報をも生成されるので、真のデータ依存関係の影響を吸収することで高速実行が可能な仮想マシンを対象とする仮想マシンコンパイラが実現される。

また、本発明に係る仮想マシンは、実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、実行対象となる仮想マシン命令列を記憶する命令記憶手段と、前記命令記憶手段から次に実行すべき仮想マシン命令を読み出す読み出し手段と、読み出された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、前記解読実行手段は、読み出された仮想マシン命令が実行制御の流れを分岐させる分岐命令であるか否かを判定する分岐命令判定部と、分岐命令であると判定される度に、その分岐命令の実行に加えて、この仮想マシンへの割込み要求の発生の有無の検出と、発生している場合の割込み処理とを実行する割込み処理部とを有することを特徴とする。

【０２３７】

これによって、１個の分岐命令が実行される度に、１回の割込み処理が実行されるので、多くの仮想マシンプログラムにおいては適度な頻度で割込み処理が繰り返されることとなり、頻繁な割込み処理が行われることによる実行速度の低下が回避されるという効果がある。

ここで、前記解読実行手段はさらに、対象とする全ての種類の仮想マシン命令に対応する実マシン命令列とこの仮想マシンへの割込み要求に対処するための実マシン命令列を記憶する実マシン命令列記憶部と、指定された実マシン命令列を実行する実行部とを備え、前記割込み処理部は、前記分岐命令判定部によって分岐命令であると判定される度に、前記実行部に、前記割込み要求に対処するための実マシン命令列を実行させた後に前記分岐命令に対応する実マシン命令列を実行させるとすることもできる。

【０２３８】

これによって、仮想マシン命令ではなく、デコードデータによって、追加処理すべき割込み処理を指定することができるので、簡易な構成によって割込み処理を行う仮想マシンが実現される。

また、本発明に係る仮想マシンは、実マシンによる制御の下で仮想マシン命令

を実行する仮想マシンであって、実行対象となる仮想マシン命令列を記憶する命令記憶手段と、前記命令記憶手段から次に実行すべき仮想マシン命令を読み出す読み出し手段と、読み出された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、前記解読実行手段は、読み出された仮想マシン命令が一定個数の仮想マシン命令の集まりをブロックとした場合の各ブロックを代表する仮想マシン命令に相当するか否かを判定するブロック判定部を備え、各ブロックを代表する仮想マシン命令であると判定される度に、その仮想マシン命令の実行に加えて、この仮想マシンへの割込み要求の発生の有無の検出と、発生している場合の割込み処理とを実行する割込み処理部とを有することを特徴とする。

【０２３９】

これによって、一定数の仮想マシン命令が実行される度に、１回の割込み処理が実行されるので、割込み処理を発生させる頻度についての制御が可能となり、頻繁な割込み処理が行われることによる実行速度の低下が回避される。

ここで、前記解読実行手段はさらに、対象とする全ての種類の仮想マシン命令に対応する実マシン命令列とこの仮想マシンへの割込み要求に対処するための実マシン命令列を記憶する実マシン命令列記憶部と、指定された実マシン命令列を実行する実行部とを備え、前記ブロック判定部は、仮想マシン命令が読み出される度に、それまでに読み出された仮想マシン命令の総数が一定値の倍数に一致するか否か比較し、一致する場合に、前記仮想マシン命令は各ブロックを代表する仮想マシン命令に相当すると判定し、前記割込み処理部は、前記分岐命令判定部によって各ブロックを代表する仮想マシン命令であると判定される度に、前記実行部に、前記割込み要求に対処するための実マシン命令列を実行させた後に前記仮想マシン命令に対応する実マシン命令列を実行させるとすることもできる。

【０２４０】

これによって、仮想マシン命令ではなく、デコードデータによって、追加処理すべき割込み処理を指定することができるので、簡易な構成によって割込み処理を行う仮想マシンが実現される。

また、本発明に係る仮想マシンは、実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、実マシン命令からなる複数のサブプログラムを

記憶する実マシンプログラム記憶手段と、実行対象となる仮想マシン命令列を記憶する第１領域と、前記実マシンプログラム記憶手段における各サブプログラムへのポインタを記憶する第２領域とを含む命令記憶手段と、前記命令記憶手段の第１領域から次に実行すべき仮想マシン命令を読み出す読み出し手段と、読み出された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、前記解読実行手段は、読み出された仮想マシン命令が前記第２領域に実行制御を移す命令であるか否かを判定する領域判定部と、前記第２領域に実行制御を移す命令であると判定された場合に、その移動先の第２領域の箇所に格納された前記ポインタが示す前記サブプログラムを実行するアドレス変換実行部とを有することを特徴とする。

【０２４１】

これによって、仮想マシン命令に代えて実マシン命令を実行させるか否かは、仮想マシンのメモリマップにおける領域によってのみ決定されるので、関数ごとに仮想マシン関数として実行させたり実マシン関数に置き換えて実行させたりする切り替え設定が容易となり、アーキテクチャの異なる複数の実マシンを対象として容易にネイティブコード化をしておくことが可能な仮想マシンが実現される。

【０２４２】

ここで、前記命令記憶手段における第１領域と第２領域とは、一定のアドレス値を境界とする隣接した記憶位置に設けられ、前記領域判定部は、読み出された仮想マシン命令がサブプログラムを呼び出す命令である場合に、その呼び出し先アドレスと前記境界との大小関係を比較することによって前記判定を行うとすることもできる。

【０２４３】

これによって、上記境界の位置を変更するだけで、仮想マシン関数への呼び出しをそのまま実行させたり実マシン関数に置き換えて実行させたりする制御が可能となるので、複数の異なるアーキテクチャの実マシンを対象とする場合であっても、そのために全体としての実行速度を低下させることなく個々の実マシンの環境に応じた実行速度の高速化が可能な仮想マシンが実現される。

【０２４４】

また、本発明に係る仮想マシンは、実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、実行対象となる仮想マシン命令列を記憶する命令記憶手段と、前記命令記憶手段から次に実行すべき仮想マシン命令を読み出す読み出し手段と、読み出された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、前記命令記憶手段は、前記仮想マシン命令列を構成する基本ブロックそれぞれに対応する命令ブロックの集まりからなり、各命令ブロックは、その命令ブロックの前記命令記憶手段における先頭位置を特定するための識別子を格納した識別子領域と、対応する基本ブロックの非分岐命令だけを格納した非分岐命令領域と、対応する基本ブロックの分岐命令を格納した分岐命令領域とを含み、前記分岐命令領域に格納された分岐命令は、その分岐先が前記識別子によって指定され、前記解読実行手段は、読み出された仮想マシン命令が前記分岐命令である場合には、その分岐先として指定されている識別子に対応する命令ブロックの非分岐命令領域の先頭に実行制御を分岐させることを特徴とする。

【０２４５】

これによって、各命令ブロックへの入口は先頭の１箇所だけとなることが保証されるので、その場式コンパイラでの分岐先命令のアドレス解析処理は単純化され、コンパイル時間が短縮される。また命令キャッシングを命令ブロック単位で行うことで、キャッシュ境界に伴う判定処理も簡略化され、仮想マシンにキャッシュを導入した場合の実行効率の低下が従来よりも抑えられるという効果がある。

【０２４６】

ここで、前記解読実行手段は、次に読み出すべき仮想マシン命令が属する命令ブロックの識別子を格納する識別子レジスタと、その仮想マシン命令のその命令ブロックにおける相対的な記憶位置を示すオフセットを格納するオフセットカウンタとからなるプログラムカウンタを備え、前記読み出し手段は、前記プログラムカウンタに格納された識別子とオフセットに基づいて前記仮想マシン命令を読み出し、前記解読実行手段は、読み出された仮想マシン命令が前記分岐命令であ

る場合には、その分岐先として指定されている識別子を前記識別子レジスタに書き込むと共に前記オフセットカウンタを初期値にリセットし、読み出された仮想マシン命令が前記非分岐命令である場合には、前記オフセットカウンタを増加させ、前記読み出し手段は、前記解読実行手段によって更新されたプログラムカウンタの識別子とオフセットに基づいて次に実行すべき仮想マシン命令を読み出すこととすることもできる。

【０２４７】

これによって、各命令ブロックは識別子レジスタの値のみによって特定され、各仮想マシン命令の相対位置はオフセットカウンタのみによって特定されるので、従来のセグメント方式によるアドレス変換技術を利用することができる。

また、前記解読実行手段はさらに、対象とする全ての種類の仮想マシン命令それぞれに対応する実マシン命令列を記憶する実マシン命令列記憶部を備え、前記命令記憶手段の各命令ブロックはさらに、その命令ブロックの非分岐命令領域及び分岐命令領域に格納された仮想マシン命令それぞれに対応する実マシン命令列を前記実マシン命令列記憶部に記憶された実マシン命令列から特定するためのデコードデータを含むデコードデータ列を格納したデコードデータ列領域を含み、前記読み出し手段は、デコードデータ列が格納された命令ブロックについては前記仮想マシン命令に代えて前記デコードデータを読み出し、デコードデータ列が格納されていない命令ブロックについては仮想マシン命令を読み出した後に、その仮想マシン命令に対応する実マシン命令列を前記実マシン命令列記憶部に記憶された実マシン命令列から特定するためのデコードデータを生成し、前記解読実行手段は、前記読み出し手段から読み出し又は生成されたデコードデータによって特定される実マシン命令列を前記実マシン命令列記憶部から読み出して実行するとすることもできる。

【０２４８】

これによって、仮想マシンプログラムを命令ブロック単位に分割して得られる上記効果に加えて、デコードデータ列が格納された命令ブロックについては、仮想マシン命令に代えてデコードデータが直接に実行されるので、解読時間が不要となり、仮想マシンの実行速度は向上されるという効果がある。

また、前記命令記憶手段の各命令ブロックはさらに、その命令ブロックのデコードデータ列領域に前記デコードデータが格納されているか否かを示すフラグを格納したフラグ領域を含み、前記解読実行手段はさらに、分岐命令を実行した場合には、その分岐先となる命令ブロックのフラグ領域に格納されたフラグを読み出して保持するカレントフラグ記憶部を備え、前記読み出し手段は、前記カレントフラグ記憶部に保持されたフラグに基づいて、デコードデータの読み出し又は仮想マシン命令の読み出しを行うとすることもできる。

【０２４９】

これによって、命令ブロックごとにデコードデータ列が格納されているか否かのフラグは、その命令ブロックから読み出されて仮想マシンに保持されるので、その命令ブロックの実行においては、仮想マシン命令ごとに命令ブロック中のフラグを参照する必要はなくなる。

また、前記命令記憶手段の各命令ブロックはさらに、その命令ブロックのデコードデータ列領域に前記デコードデータが格納されているか否かを示すフラグを格納したフラグ領域を含み、前記解読実行手段はさらに、分岐命令を実行した場合にその分岐先となる命令ブロックのフラグ領域に格納されたフラグを参照することにより、その命令ブロックにはデコードデータが格納されていないと判定したときには、その命令ブロックに格納されている仮想マシン命令列を読み出して対応するデコードデータ列に変換し、その命令ブロックのデコードデータ列領域に書き込むデコードデータ列書込み部を備えるとすることもできる。

【０２５０】

これによって、デコードデータ列を格納していない命令ブロックであっても、初めての実行時にデコードデータ列が生成されて格納されるので、ループ処理等のように繰り返して実行される場合には、２回目以降の実行時における速度が向上される。

また、本発明に係る仮想マシンは、実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、実行対象となる仮想マシン命令列を圧縮符号で記憶する命令記憶手段と、前記命令記憶手段から次に実行すべき仮想マシン命令の圧縮符号を読み出し、対応する仮想マシン命令に復号する読み出し手段と、復

号された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、前記命令記憶手段は、前記仮想マシン命令列を構成する基本ブロックそれぞれに対応する命令ブロックの集まりからなり、各命令ブロックは、その命令ブロックの前記命令記憶手段における先頭位置を特定するための識別子を格納した識別子領域と、対応する基本ブロックの非分岐命令だけを格納した非分岐命令領域と、対応する基本ブロックの分岐命令を格納した分岐命令領域とを含み、前記分岐命令領域に格納された分岐命令は、その分岐先が前記識別子によって指定され、前記解読実行手段は、復号された仮想マシン命令が前記分岐命令である場合には、その分岐先として指定されている識別子に対応する命令ブロックの非分岐命令領域の先頭に実行制御を分岐させることを特徴とする。

【０２５１】

これによって、実行対象となる仮想マシンプログラムは、基本ブロックを単位とし、かつ、圧縮符号化されて命令記憶手段に格納され、解読実行手段によって復号化され実行されるので、圧縮符号の途中に分岐することに伴う不具合を生じることのない仮想マシンが実現される。

また、前記命令記憶手段の各命令ブロックにはさらに、その命令ブロックに格納された仮想マシン命令の圧縮符号を伸長復号するための情報テーブルであって、対応する圧縮符号と仮想マシン命令との組の集まりからなる復号テーブルを格納した復号テーブル領域を含み、前記読み出し手段は、次に実行すべき仮想マシン命令が属する命令ブロックの復号テーブルを参照しながら、前記命令記憶手段から読み出した圧縮符号を対応する仮想マシン命令に復号するとすることもできる。

【０２５２】

これによって、命令ブロックごとに復号テーブルが格納され参照されるので、命令ブロックごとに異なる種類の圧縮符号化を施した場合であっても、適正に復号されて実行されることが保証される。

また、本発明に係るその場式コンパイラは、実マシンによる制御の下で仮想マシン命令を実行する仮想マシンと共に用いられ、実行対象となる一部の仮想マシン命令列をその実行に先立って実マシン命令列に変換するその場式コンパイラで

あって、前記仮想マシン命令列を構成する仮想マシン命令それぞれについて、前記仮想マシン命令列を基本ブロックに分割した場合の各基本ブロックの先頭となる仮想マシン命令であるか否かを示すブロック先頭情報の入力を受け付けるブロック先頭情報獲得手段と、前記仮想マシン命令列を構成する仮想マシン命令それぞれについて、対応する実マシン命令列に変換する変換手段と、前記変換手段によって得られた実マシン命令列における実マシン命令であって、前記ブロック先頭情報獲得手段が受け付けたブロック先頭情報によって特定された基本ブロックの先頭となる仮想マシン命令に対応する実マシン命令が、前記実マシンによってアドレシングされ得ない位置に配置される命令であるか否かを判定する分岐違反判定手段と、前記分岐違反判定手段によって前記実マシン命令が前記位置に配置される命令であると判定された場合に、前記実マシン命令が前記位置に配置されないよう１個以上の無動作命令を前記実マシン命令列に追加挿入して出力する出力手段とを備えることを特徴とする。

【０２５３】

これによって、従来において必要とされた分岐命令の分岐先についての解析という複雑な処理を行うことなく、ジャンプ先についての境界違反を起こすことがない実マシンプログラムを高速に生成するその場式コンパイラが実現される。

ここで、前記出力手段は、前記実マシン命令が前記実マシンによってアドレシングされ得る後続位置にずらせるために必要な個数の無動作命令を前記実マシン命令列における基本ブロックの先頭位置に追加挿入するとすることもできる。

【０２５４】

これによって、生成される実マシンプログラムの実質的な内容を変化させることなく、ジャンプ命令に伴う境界違反の発生を回避したその場式コンパイラが実現される。

以上のように、本発明によって、仮想マシンの実行速度は高速化され、特に、異機種のコンピュータ同士が接続されるネットワーク環境における共有資源の迅速かつ円滑な利用を促進させる基盤技術として、その実用的価値は極めて大きい。

【図面の簡単な説明】

【図１】

実施形態１～９に係る仮想マシンシステムが動作するコンピュータのハードウエア構成図である。

【図２】

実施形態１に係る仮想マシン１００の構成を示す機能ブロック図である。

【図３】

図３（ａ）及び（ｂ）は、それぞれ先行命令情報格納部１０１及び命令格納部１０２に格納されている先行命令情報及び対応する仮想マシンコードの例を示す図である。

【図４】

デコードテーブル１０８の内容を示す図である。

【図５】

図５（ａ）及び（ｂ）は、それぞれ、上向き仮想マシン命令"Push"及び下向き仮想マシン命令"Push"に対応するマイクロプログラムのリストを示す。

【図６】

図６（ａ）及び（ｂ）は、それぞれ、上向き仮想マシン命令"Add"及び下向き仮想マシン命令"Add"に対応するマイクロプログラムのリストを示す。

【図７】

図７（ａ）及び（ｂ）は、それぞれ、上向き仮想マシン命令"Mult"及び下向き仮想マシン命令"Mult"に対応するマイクロプログラムのリストを示す。

【図８】

図８（ａ）及び（ｂ）それぞれは、図６（ａ）及び図７（ａ）に示された上向き仮想マシン命令のマイクロプログラムの後半に相当するマイクロプログラム、図６（ｂ）及び図７（ｂ）に示された下向き仮想マシン命令のマイクロプログラムの後半に相当するマイクロプログラムのリストを示す。

【図９】

本仮想マシン１００によって実行される仮想マシン命令が属する種別の変化を示す状態遷移図である。

【図１０】

デコード部１０３の動作を示すフローチャートである。

【図１１】

図１０に示されたテーブル検索（ステップ４９０７）の詳細を示すフローチャートの前半部である。

【図１２】

図１０に示されたテーブル検索（ステップ４９０７）の詳細を示すフローチャートの後半部である。

【図１３】

デコード部１０３が実行部１１０に順次出力するデコードデータ列の例を示す図である。

【図１４】

図１４（ａ）及び（ｂ）は、図１３に示されたデコードデータ列に従って実行部１１０が動作した場合における本仮想マシン１００の内部状態の変化を示す図である。

【図１５】

実マシン２０１が通常マシンであってメモリ参照を行ってから１クロック後にその参照値を使用することができる場合におけるパイプラインの流れを示す図である。、

【図１６】

実マシン２０１がスーパースカラマシンであってメモリ参照を行ってから１クロック後にその参照値を使用することができる場合におけるパイプラインの流れを示す図である。、

【図１７】

実マシン２０１が通常マシンであってメモリ参照を行ってから２クロック後にその参照値を使用することができる場合におけるパイプラインの流れを示す図である。、

【図１８】

実マシン２０１がスーパースカラマシンであってメモリ参照を行ってから２クロック後にその参照値を使用することができる場合におけるパイプラインの流れ

を示す図である。

【図１９】

実施形態１に係る仮想マシンコンパイラ３４００の構成を示す機能ブロック図である。

【図２０】

命令列変換部３４０２に入力されるソースプログラム３４０４のデータ構造を示す図である。

【図２１】

図２０における各ノードのデータ構造を示す図である。

【図２２】

命令列変換部３４０２の動作手順の概略を示すフローチャートである。

【図２３】

図２２におけるステップ５４０５の詳細を示すフローチャートである。

【図２４】

図２３におけるステップ５６１３の詳細を示すフローチャートである。

【図２５】

先行命令情報生成部３４０１の動作を示すフローチャートである。

【図２６】

関連付け部３４０３の動作を示すフローチャートである。

【図２７】

実施形態２に係る仮想マシン３５００の構成を示す機能ブロック図である。

【図２８】

デコード部３５０２の動作のうち、テーブル検索とデコードデータの出力処理の詳細を示すフローチャートである。

【図２９】

分岐命令検出部３５０５の動作を示すフローチャートである。

【図３０】

割込み命令挿入部３５０６の動作を示すフローチャートである。

【図３１】

実施形態３に係る仮想マシン３６００の構成を示す機能ブロック図である。

【図３２】

ブロック化部３６０５の動作を示すフローチャートである。

【図３３】

実施形態４に係る仮想マシン３７００の構成を示す機能ブロック図である。

【図３４】

命令格納部３７０１のメモリマップを示す図である。

【図３５】

図３４に示された実マシン関数テーブル６５０２の構造を示す図である。

【図３６】

実行部３７１０の動作を示すフローチャートである。

【図３７】

命令格納部３７０１のメモリマップの変形例を示す図である。

【図３８】

実施形態５に係る仮想マシン３８００の構成を示す機能ブロック図である。

【図３９】

命令格納部３８０１における仮想マシンプログラムの格納状態の例を示す図である。

【図４０】

図３９に示された仮想マシンプログラムの制御フローを示す図である。

【図４１】

ＰＣ３８０４によるアドレシングのフォーマットを示す図である。

【図４２】

実行部３８１０における分岐先変換部３８１１の動作を示すフローチャートである。

【図４３】

図３９に示された仮想マシンプログラムにおける論理的なアドレスと識別子とを物理的なアドレスに置き換えた図である。

【図４４】

実施形態５に係る仮想マシンコンパイラ７６６０の構成示すブロック図である。

【図４５】

分岐先変更テーブル７６６３ａの構造を示す図である。

【図４６】

ブロック化部７６６３の動作を示すフローチャートである。

【図４７】

図４６におけるステップ７６０７の詳細を示すフローチャートである。

【図４８】

図４６におけるステップ７７０４の詳細を示すフローチャートである。

【図４９】

図４６におけるステップ７６０９の詳細を示すフローチャートである。

【図５０】

命令ブロック単位でキャッシュさせた場合のＰＣ３８０４、命令ブロック格納領域３８５２ａ～３８５２ｄ及びキャッシュテーブル８４０４の関係を示す図である。

【図５１】

命令ブロック単位でキャッシュさせた場合の実行部３８１０による分岐命令の実行処理を示すフローチャートである。

【図５２】

実施形態６に係る仮想マシン３９００の構成を示す機能ブロック図である。

【図５３】

命令格納部３９０１に格納されている仮想マシンプログラムの格納状態の例を示す図である。

【図５４】

デコード部３９０２の動作を示すフローチャートである。

【図５５】

実行部３９１０の動作を示すフローチャートである。

【図５６】

実行部３９１０が分岐命令を実行した場合のデコード部３９０２に対する制御を示すフローチャートである。

【図５７】

実施形態７に係る仮想マシン４０００の構成を示す機能ブロック図である。

【図５８】

本仮想マシン４０００が分岐命令を実行した場合におけるデコード命令列書込み部４００８、カレントフラグ読出制御部３９１２及び分岐先変換部３８１１における動作を示すフローチャートである。

【図５９】

図５８におけるステップ９１１０の詳細を示すフローチャートである。

【図６０】

実行部３９１０から見たデコード部４００２の動作を示すフローチャートである。

【図６１】

実施形態８に係る仮想マシン４１００の構成を示す機能ブロック図である。

【図６２】

図６２（ａ）は、展開情報格納領域４１５７ａ～４１５７ｄに格納された復号化テーブルの例を示し、図６２（ｂ）は、図６２（ａ）に示された復号化テーブルにおける符号の規則性を示す図である。

【図６３】

命令格納部４１０１に格納されている仮想マシンプログラムの格納状態の例を示す図である。

【図６４】

デコード部４１０２の動作を示すフローチャートである。

【図６５】

図６４におけるステップ９６０２の詳細を示すフローチャートである。

【図６６】

実施形態９に係るその場式コンパイラ４３００を含むコンパイラシステム全体の構成を示す機能ブロック図である。

【図６７】

ブロック先頭情報生成部４３２１ａの動作を示すフローチャートである。

【図６８】

実マシン命令変換部４３０１、分岐位置補正部４３０２及び実マシンアドレス保持部４３０３の動作を示すフローチャートである。

【図６９】

ブロック先頭情報生成部４３２１ａによって生成されるブロック先頭情報、分岐位置補正部４３０２によって生成される実マシン命令"Nop"のタイミング、その他の関連情報を示すテーブルである。

【図７０】

本発明に係る仮想マシンが実行する仮想マシン命令のフォーマットの変形例を示す図である。

【図７１】

従来のスタック型仮想マシン４４００の構成を示す機能ブロック図である。

【図７２】

従来及び本発明に係る仮想マシンの命令セットを説明するための図である。

【図７３】

図７１に示されたデコードテーブル４４０６の内容を示す図である。

【図７４】

図７１に示されたマイクロプログラム記憶部４４１１に格納されているマイクロプログラムのリストを示す図である。

【図７５】

従来及び本発明に係る実マシン命令の意味を示す図である。

【図７６】

図７１に示されたデコード部４４０２の動作を示すフローチャートである。

【図７７】

図７６のフローチャートにおけるステップ４５０６の詳細を示すフローチャートである。

【図７８】

デコード部４４０２からのデコードデータがバッファを介して実行部４４１０に渡される場合におけるデコード部４４０２の動作を示すフローチャートである。

【図７９】

図７１に示された実行部４４１０の動作を示すフローチャートである。

【図８０】

図８０（ａ）は、サンプルプログラムのリストを示し、図８０（ｂ）は、そのプログラムが指示する演算式「2*(3+4)」を示し、図８０（ｃ）は、デコード部４４０２から順次出力されるデコードデータを示す図である。

【図８１】

図８０（ｃ）に示されたデコードデータ列に従って実行部４４１０が動作した場合における従来の仮想マシンの内部状態の変化を示す図である。

【図８２】

TOS変数を用いる従来の仮想マシンにおけるマイクロプログラムのリストを示す図である。

【図８３】

図８２（ａ）～（ｄ）に示されたマイクロプログラムを備える従来の仮想マシンによって図８０（ａ）の仮想マシンプログラムが実行された場合の仮想マシンの内部状態の変化を示す図である。

【図８４】

パイプラインの各ステージに対する省略記号を説明する図である。

【図８５】

通常マシンのパイプラインの理想的な流れを示す図である。

【図８６】

スーパースカラマシンのパイプラインの理想的な流れを示す図である。

【図８７】

通常マシンにおいて、パイプラインにハザードが生じた場合のパイプラインの流れを示す図である。

【図８８】

スーパースカラマシンにおいて、パイプラインにハザードが生じた場合のパイプラインの流れを示す図である。

【図８９】

図８７の場合において、メモリ参照が行われてからその値が次に使用されるまでに２クロックの時間を必要とするときのパイプラインの流れを示す図である。

【図９０】

図８８の場合において、メモリ参照が行われてからその値が次に使用されるまでに２クロックの時間を必要とするときのパイプラインの流れを示す図である。

【図９１】

通常マシンにおいて、命令Ａ及び命令Ａ２がレジスタでジャンプ先を指定する命令である場合のパイプラインの流れを示す図である。

【図９２】

スーパースカラマシンにおいて、命令Ａ及び命令Ａ２がレジスタでジャンプ先を指定する命令である場合のパイプラインの流れを示す図である。

【図９３】

第１の従来技術に係る仮想マシンが図８０（ａ）に示された仮想マシンプログラムを実行した場合であって、メモリ参照が行われてからその値が次に使用されるまでに１クロックの時間で済む場合における通常マシンのパイプラインの流れを示す。

【図９４】

図９３と同様の場合であって、スーパースカラマシンのパイプラインの流れを示す。

【図９５】

メモリ参照が行われてからその値が次に使用されるまでに２クロックの時間を必要とする場合における通常マシンのパイプラインの流れを示す。

【図９６】

図９５と同様の場合であって、スーパースカラマシンのパイプラインの流れを示す。

【図９７】

サンプル仮想マシンプログラムのリストを示す図である。

【図９８】

図９７に示されたサンプル仮想マシンプログラムのフローチャートである。

【図９９】

従来のその場式コンパイルにおいて使用される変換テーブルである。

【図１００】

図９９に示された変換テーブルを用いて図９７に示された仮想マシンプログラムをコンパイルした場合に得られる実マシンプログラムのコード配置の様子を図である。

【図１０１】

圧縮符号化テーブルの例を示す図である。

図１０１（ａ）は、圧縮符号化テーブルの例を示し、図１０１（ｂ）は、誤って復号化される場合の例を示す図である。

【図１０２】

キャッシュメモリを備える従来の仮想マシンにおいて生じうる問題点を説明するための図である。

【図１０３】

図９７に示された仮想マシンプログラムがキャッシュメモリに置かれた場合における各キャッシュブロックを区切る境界線Ａ，Ｂ，Ｃを示す図である。

【符号の説明】

１００　仮想マシン

１０１　先行命令情報格納部

１０２　命令格納部

１０３　デコード部

１０４　先行命令情報読込み部

１０５　命令読込み部

１０６　検索部

１０７　ＰＣ

１０８　デコードテーブル

１１０　実行部
１１１　マイクロプログラム記憶部
１１２　ＳＰ
１２０　スタック
１２１　ＴＯＳ変数
１２２　ＳＯＳ変数
１２３　メモリスタック
２００　コンピュータ
２０１　実マシン
２０２　メモリ
２０３　キーボード
２０４　マウス
２０５ａ～２０５ｃ　内部バス
２０６　ディスプレイ
２０７　ハードディスク
２０８　ネットワークカード
３４００　仮想マシンコンパイラ
３４０１　先行命令情報生成部
３４０２　命令列変換部
３４０３　関連付け部
３４０４　ソースプログラム
３４０５　出力プログラム
３５００　仮想マシン
３５０２　デコード部
３５０５　分岐命令検出部
３５０６　割込み命令挿入部
３５０７　割込み状態格納部
３５１０　割込み制御部
３５１５　実行部

３５１６　割込み処理プログラム
３６００　仮想マシン
３６０５　ブロック化部
３６１０　割込み制御部
３７００　仮想マシン
３７０１　命令格納部
３７０４　領域判定部
３７０５　アドレス変換部
３７０６　実マシン関数記憶部
３７１０　実行部
３８００　仮想マシン
３８０１　命令格納部
３８０２　デコード部
３８０４　ＰＣ
３８０４ａ　識別子セグメントレジスタ
３８０４ｂ　オフセットカウンタ
３８１０　実行部
３８１１　分岐先変換部
３８５２ａ～３８５２ｄ　命令ブロック格納領域
３８５３ａ～３８５３ｄ　識別子格納領域
３８５４ａ～３８５４ｄ　非分岐命令格納領域
３８５５ａ～３８５５ｄ　分岐命令格納領域
３９００　仮想マシン
３９０１　命令格納部
３９０２　デコード部
３９０３　命令読込み部
３９０７　カレントフラグ記憶部
３９１０　実行部
３９１２　カレントフラグ読出制御部

３９５２ａ～３９５２ｄ　命令ブロック格納領域
３９５３ａ～３９５３ｄ　識別子格納領域
３９５４ａ～３９５４ｄ　非分岐命令格納領域
３９５５ａ～３９５５ｄ　分岐命令格納領域
３９５６ａ～３９５６ｄ　デコード命令列格納領域
４０００　仮想マシン
４００２　デコード部
４００８　デコード命令列書込み部
４１００　仮想マシン
４１０１　命令格納部
４１０２　デコード部
４１０３　命令読込み部
４１０３ａ　仮想マシン命令展開部
４１５２ａ～４１５２ｄ　命令ブロック格納領域
４１５３ａ～４１５３ｄ　識別子格納領域
４１５４ａ～４１５４ｄ　非分岐命令格納領域
４１５５ａ～４１５５ｄ　分岐命令格納領域
４１５６ａ～４１５６ｄ　デコード命令列格納領域
４１５７ａ～４１５７ｄ　展開情報格納領域
４１６４ａ～４１６４ｄ　圧縮仮想マシンコード領域
４３００　その場式コンパイラ
４３０１　実マシン命令変換部
４３０２　分岐位置補正部
４３０３　実マシンアドレス記憶部
４３１１　実マシン命令列
４３２０　仮想マシンコンパイラ
４３２１　出力部
４３２１ａ　ブロック先頭情報生成部
６５０１　仮想マシンプログラム領域

６５０２　実マシン関数テーブル領域

６７０１　メモリ属性

７６５０　ソースプログラム

７６５１　命令ブロック集合

７６６０　仮想マシンコンパイラ

７６６１　中間命令列変換部

７６６２　生成部

７６６３　ブロック化部

７６６３ａ　分岐先変更テーブル

７６６４　仮想マシンプログラム

８４０２　命令キャッシュ

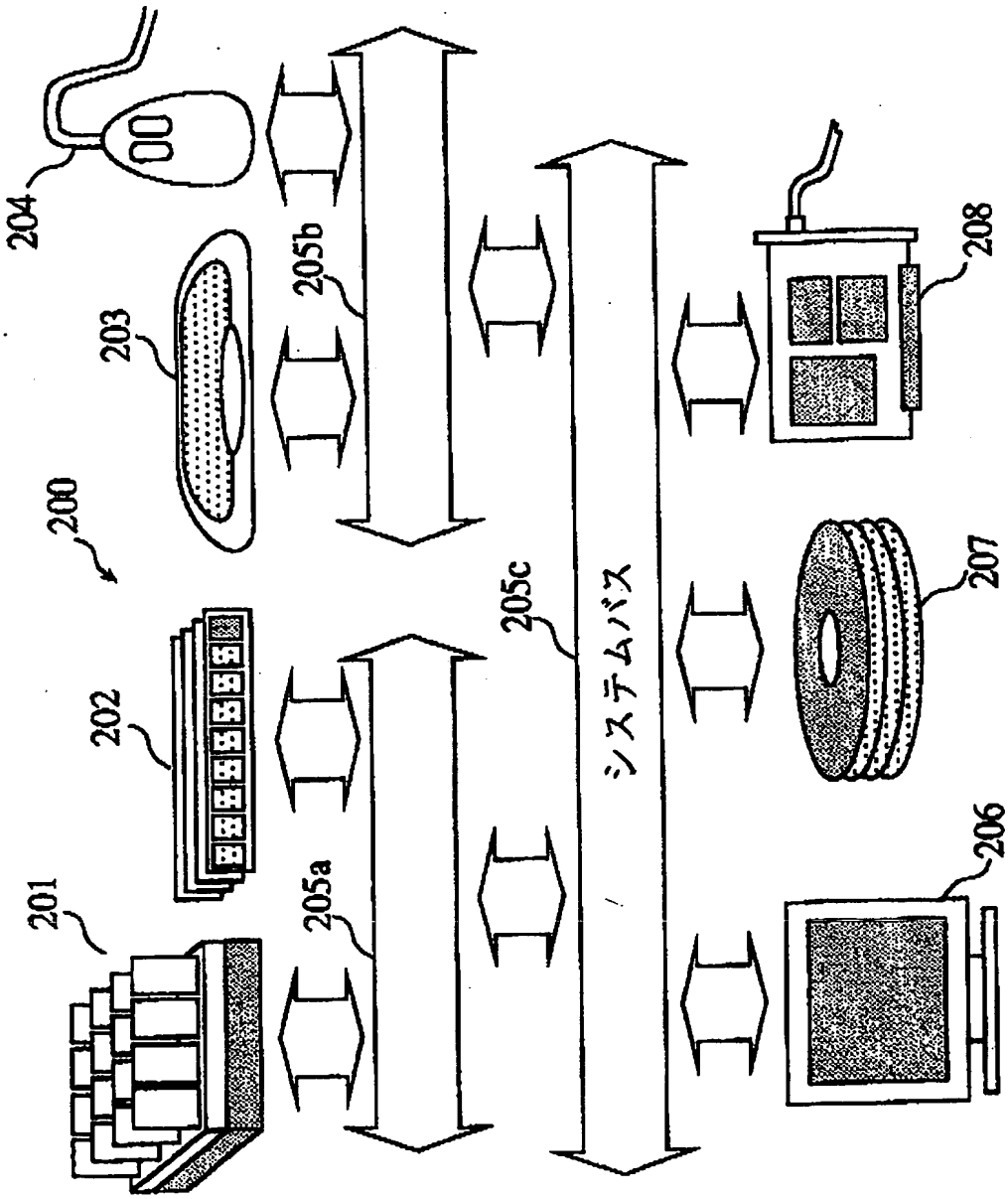
８４０４　キャッシュテーブル

８６０５ａ～８６０５ｄ　フラグ領域

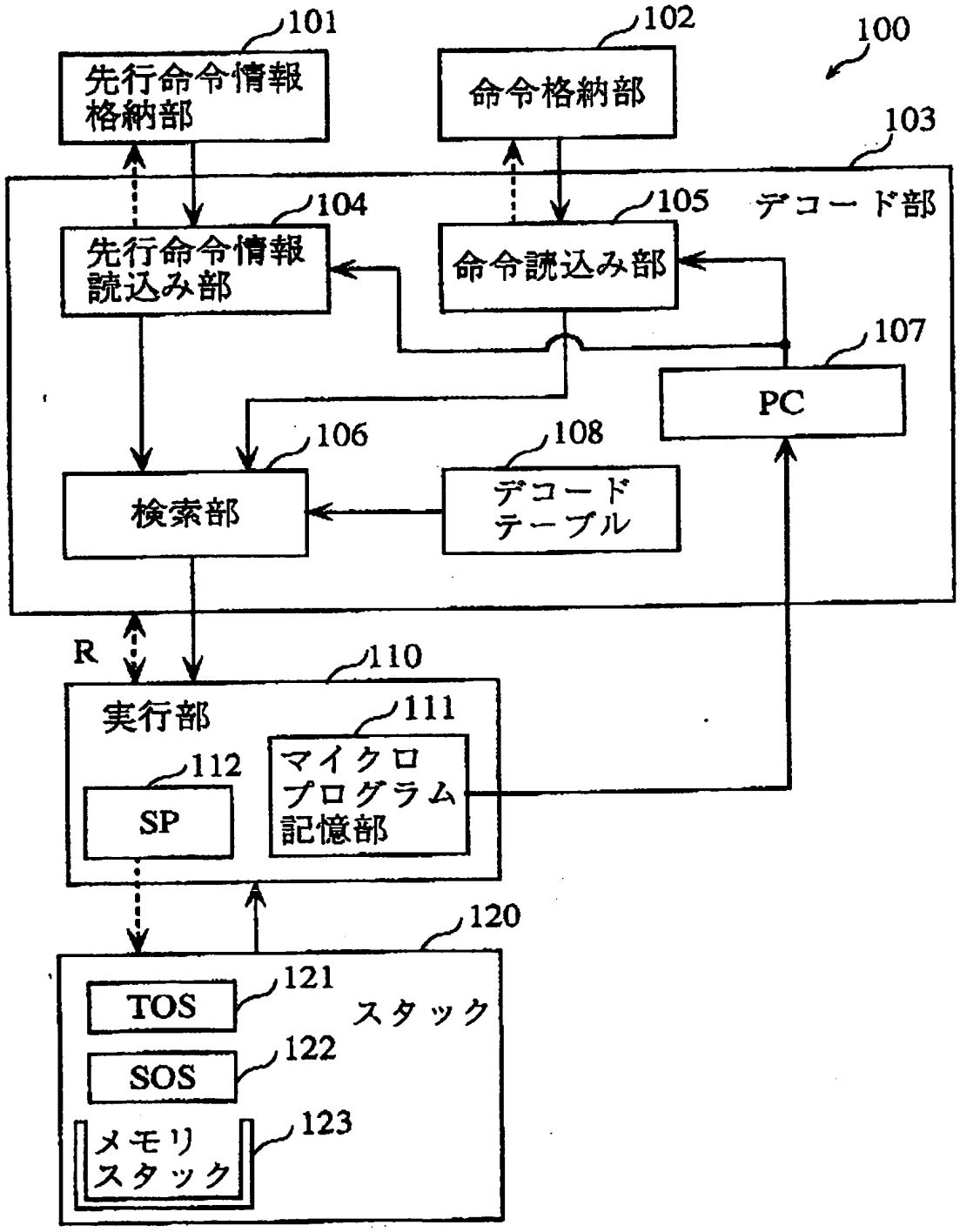
８６０７ａ～８６０７ｄ　実マシンコード領域

【書類名】　図面

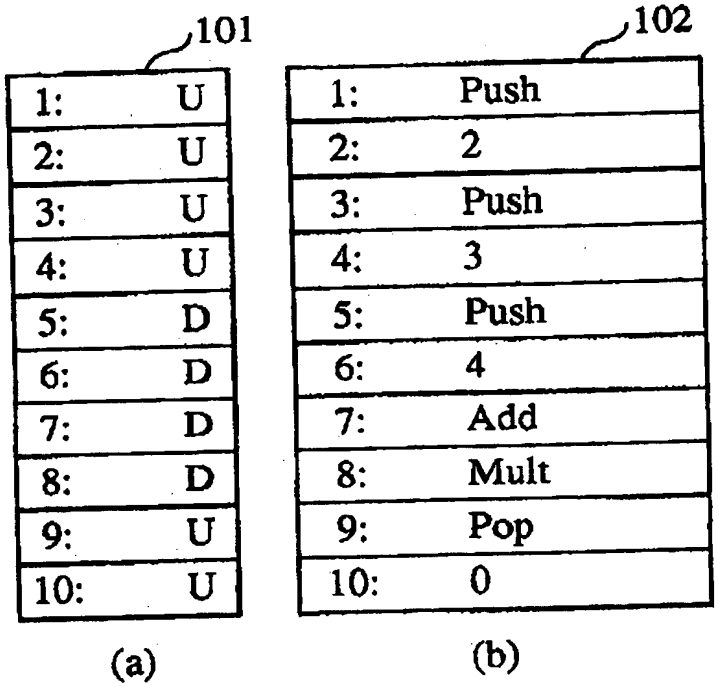
【図１】



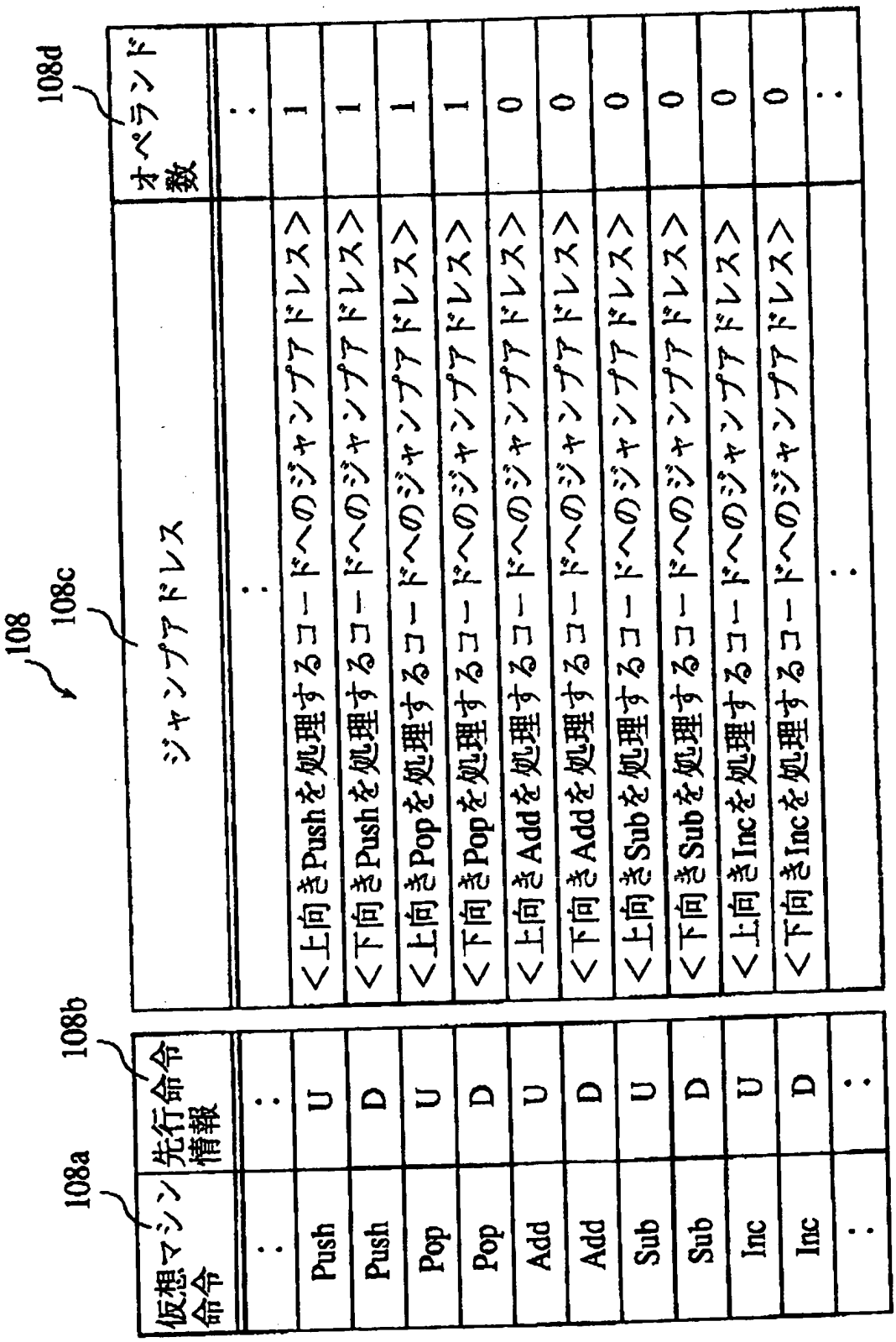
【図２】


101
先行命令情報格納部
102
命令格納部
100
103
デコード部
104
先行命令情報読込み部
105
命令読込み部
107
PC
106
検索部
108
デコードテーブル
R
110
実行部
111
マイクロプログラム記憶部
112
SP
120
スタック
121
TOS
122
SOS
123
メモリスタック

【図３】


101
1: U
2: U
3: U
4: U
5: D
6: D
7: D
8: D
9: U
10: U
(a)
102
1: Push
2: 2
3: Push
4: 3
5: Push
6: 4
7: Add
8: Mult
9: Pop
10: 0
(b)

【図４】


108
108a
108b
108c
108d
仮想マシン命令
先行命令情報
ジャンプアドレス
オペランド数
Push U <上向きPushを処理するコードへのジャンプアドレス> 1
Push D <下向きPushを処理するコードへのジャンプアドレス> 1
Pop U <上向きPopを処理するコードへのジャンプアドレス> 1
Pop D <下向きPopを処理するコードへのジャンプアドレス> 0
Add U <上向きAddを処理するコードへのジャンプアドレス> 0
Add D <下向きAddを処理するコードへのジャンプアドレス> 0
Sub U <上向きSubを処理するコードへのジャンプアドレス> 0
Sub D <下向きSubを処理するコードへのジャンプアドレス> 0
Inc U <上向きIncを処理するコードへのジャンプアドレス> 0
Inc D <下向きIncを処理するコードへのジャンプアドレス> 0

【図５】

(a)

仮想マシン命令(上向き)"Push"のマイクロプログラム		
1:Load	r4,r0	;TOSレジスタ(0番)の値を、SOSレジスタ ;(4番)にコピーする
2:Load	r0,[r2]	;オペランドを取り出し、 ;TOSレジスタに読み込む
3:Inc	r2	;仮想マシンのPCを一つ増やし、 ;次の命令を読み込めるようにする
4:Inc	r3	;仮想マシンのSPの値を増やす
5:Store	[r3],r4	;SOSレジスタの値を、スタックに格納する
6:Load	r1,[r2]	;仮想マシンのPCの位置の仮想 ;マシン命令(ジャンプアドレス)を ;レジスタ1番に読み込む
7:Inc	r2	;仮想マシンのPCの値を1増やす
8:Jmp	r1	;レジスタ1番で示される位置に ;無条件ジャンプする

(b)

仮想マシン命令(下向き)"Push"のマイクロプログラム		
1:Load	r4,r0	;TOSレジスタの値を、SOSレジスタに ;コピーする
2:Load	r0,[r2]	;オペランドを取り出し、 ;TOSレジスタに読み込む
3:Inc	r2	;仮想マシンのPCを一つ増やし、 ;次の命令を読み込めるようにする
6:Load	r1,[r2]	;仮想マシンのPCの位置の仮想 ;マシン命令(ジャンプアドレス)を ;レジスタ1番に読み込む
7:Inc	r2	;仮想マシンのPCの値を1増やす
8:Jmp	r1	;レジスタ1番で示される位置に ;無条件ジャンプする

【図６】

(a)

仮想マシン命令(上向き)"Add"のマイクロプログラム
1:Add r0,r0,r4 ;TOSレジスタと、SOSレジスタを加え、 ;結果をTOSレジスタに代入する ＜上向きの仮想マシンジャンプコード＞

(b)

仮想マシン命令(下向き)"Add"のマイクロプログラム
1:Add r0,r0,r4 ;TOSレジスタと、SOSレジスタを加え、 ;結果をTOSレジスタに代入する ＜下向きの仮想マシンジャンプコード＞

【図７】

(a)

仮想マシン命令(上向き)"Mult"のマイクロプログラム
1:Mult r0,r0,r4 ;TOSレジスタと、SOSレジスタを掛け、 ;結果をTOSレジスタに代入する ＜上向きの仮想マシンジャンプコード＞

(b)

仮想マシン命令(下向き)"Mult"のマイクロプログラム
1:Mult r0,r0,r4 ;TOSレジスタと、SOSレジスタを掛け、 ;結果をTOSレジスタに代入する ＜下向きの仮想マシンジャンプコード＞

【図８】

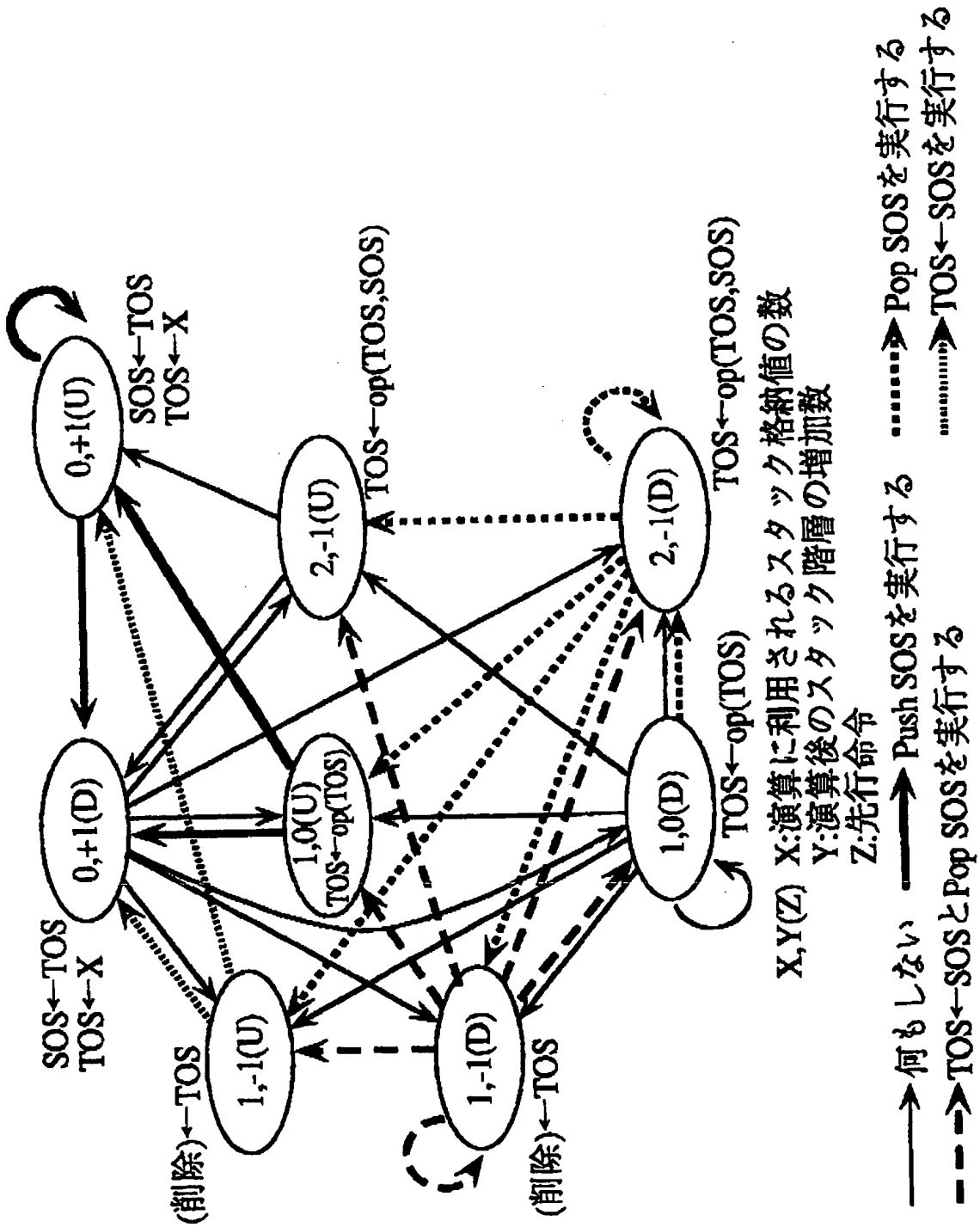
(a)

上向きの仮想マシンジャンプコード	
1:Load r1,[r2]	;仮想マシンのPCの位置の仮想 ;マシン命令(ジャンプアドレス)を ;レジスタ1番に読み込む
2:Inc r2	;仮想マシンのPCの値を1増やす
3:Jmp r1	;レジスタ1番で示される位置に ;無条件ジャンプする

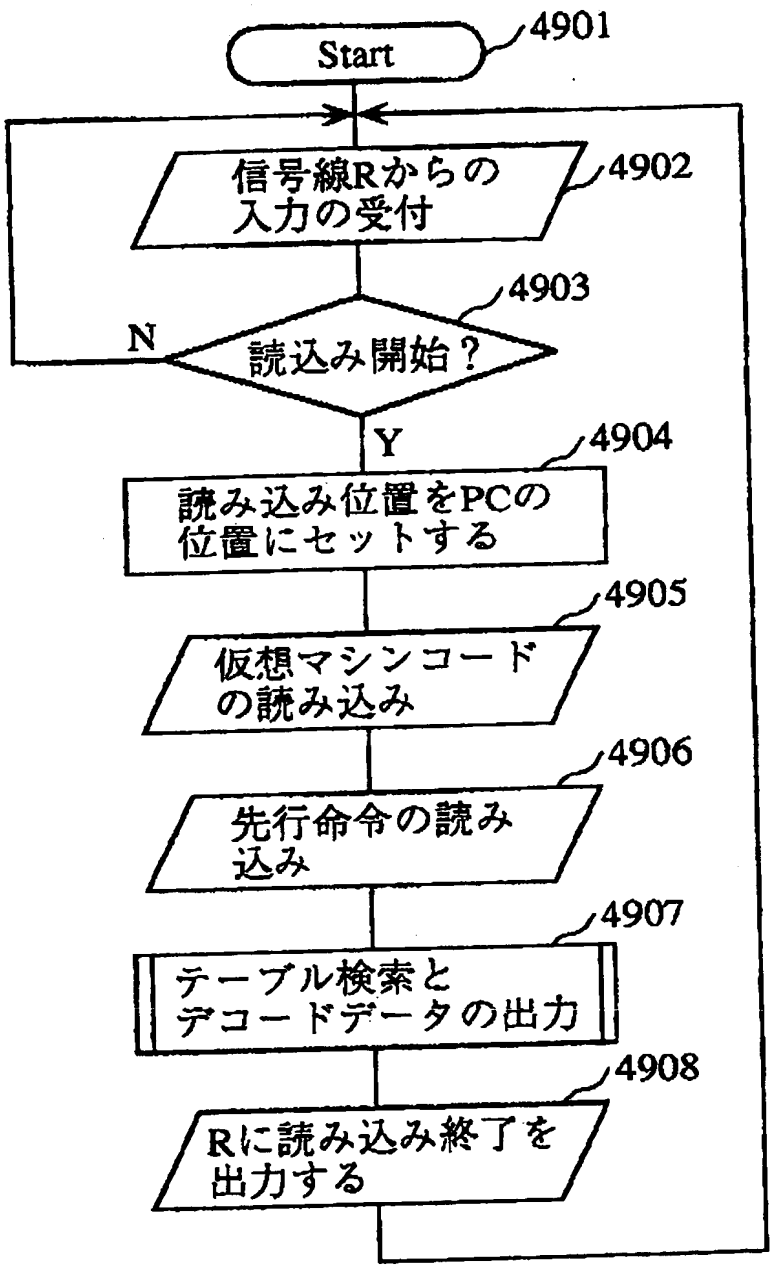
(b)

下向きの仮想マシンジャンプコード	
1:Load r1,[r2]	;仮想マシンのPCの位置の仮想 ;マシン命令(ジャンプアドレス)を ;レジスタ1番に読み込む
2:Load r4,[r3]	;スタックから値を取り出し、SOSレジスタに ;コピーする
3:Inc r2	;仮想マシンのPCの値を1増やす
4:Dec r3	;仮想マシンのSPの値を1減らす
5:Jmp r1	;レジスタ1番で示される位置に ;無条件ジャンプする

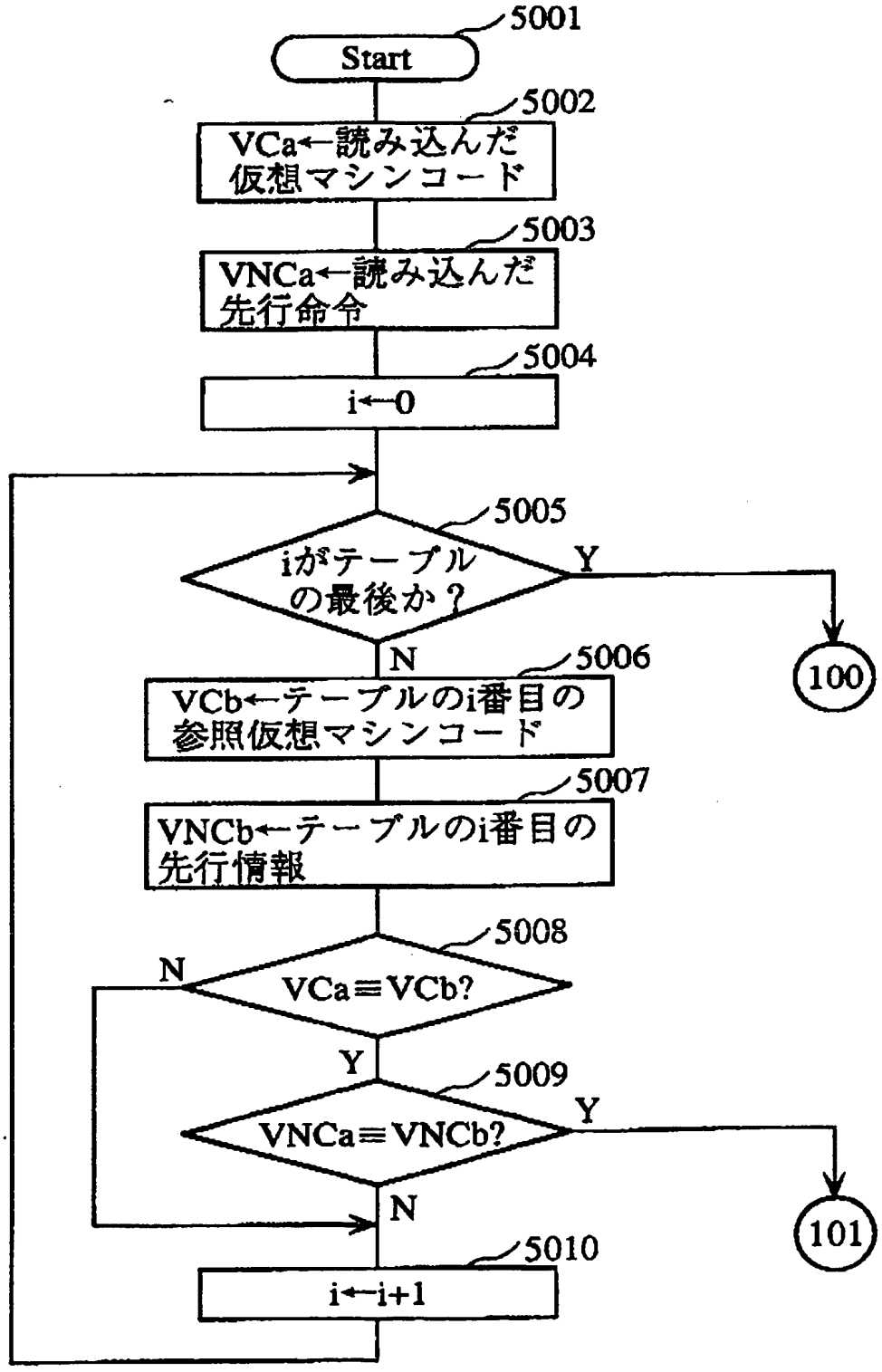
【図９】


0,+1(U)
SOS←TOS
TOS←X
2,-1(U)
TOS←op(TOS,SOS)
2,-1(D)
TOS←op(TOS,SOS)
0,+1(D)
SOS←TOS
TOS←X
1,0(U)
TOS←op(TOS)
1,0(D)
TOS←op(TOS)
1,-1(U)
(削除)←TOS
1,-1(D)
(削除)←TOS
X,Y(Z) X:演算に利用されるスタック格納値の数
Y:演算後のスタック階層の増加数
Z:先行命令
何もしない
Push SOSを実行する
Pop SOSを実行する
TOS←SOSとPop SOSを実行する
TOS←SOSを実行する

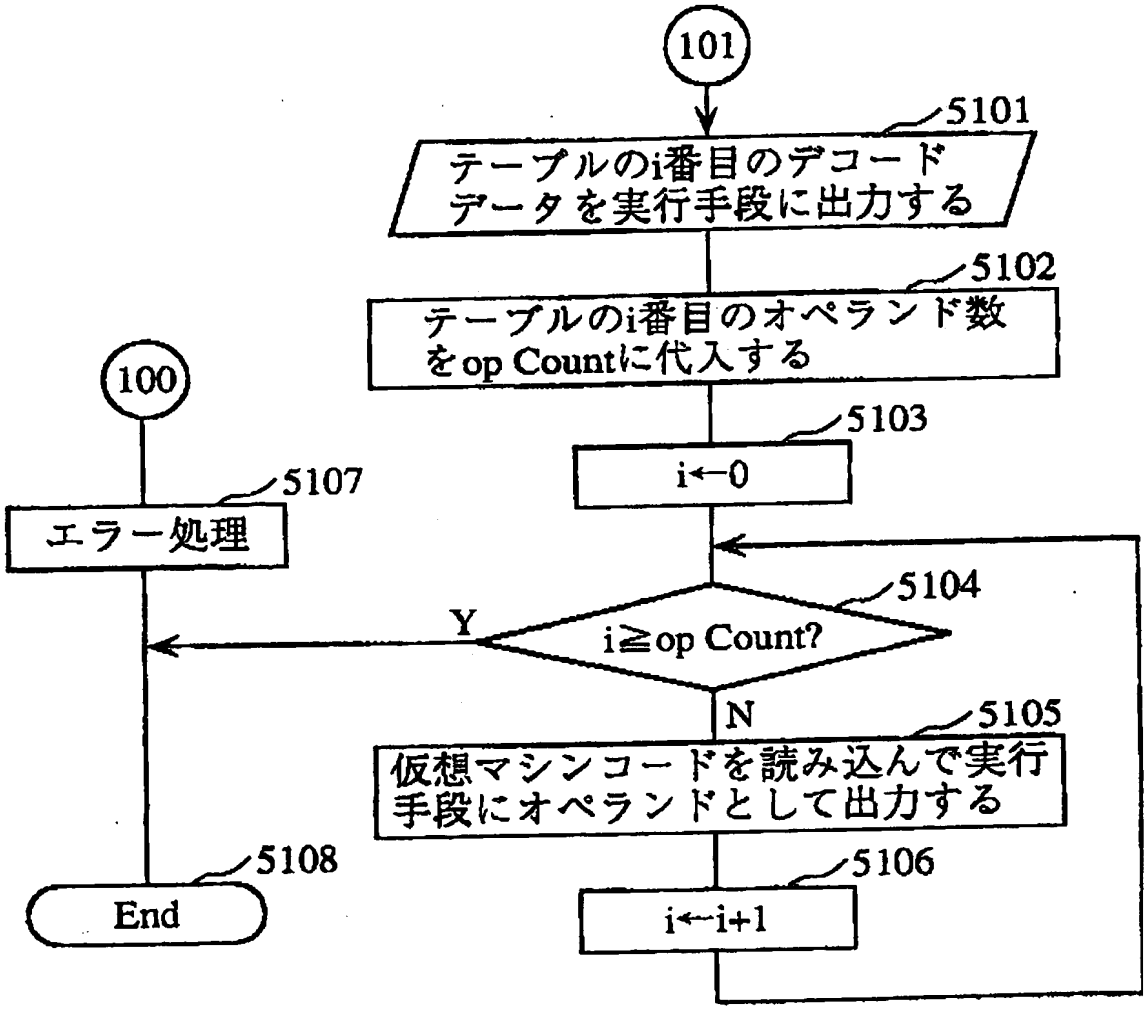
【図１０】


Start
4901
信号線Rからの
入力の受付
4902
4903
N
読込み開始？
Y
4904
読み込み位置をPCの
位置にセットする
4905
仮想マシンコード
の読み込み
4906
先行命令の読み
込み
4907
テーブル検索と
デコードデータの出力
4908
Rに読み込み終了を
出力する

【図１１】


5001
Start
5002
VCa←読み込んだ仮想マシンコード
5003
VNCa←読み込んだ先行命令
5004
i←0
5005
iがテーブルの最後か？
Y
N
100
5006
VCb←テーブルのi番目の参照仮想マシンコード
5007
VNCb←テーブルのi番目の先行情報
5008
VCa≡VCb?
N
Y
5009
VNCa≡VNCb?
Y
N
101
5010
i←i+1

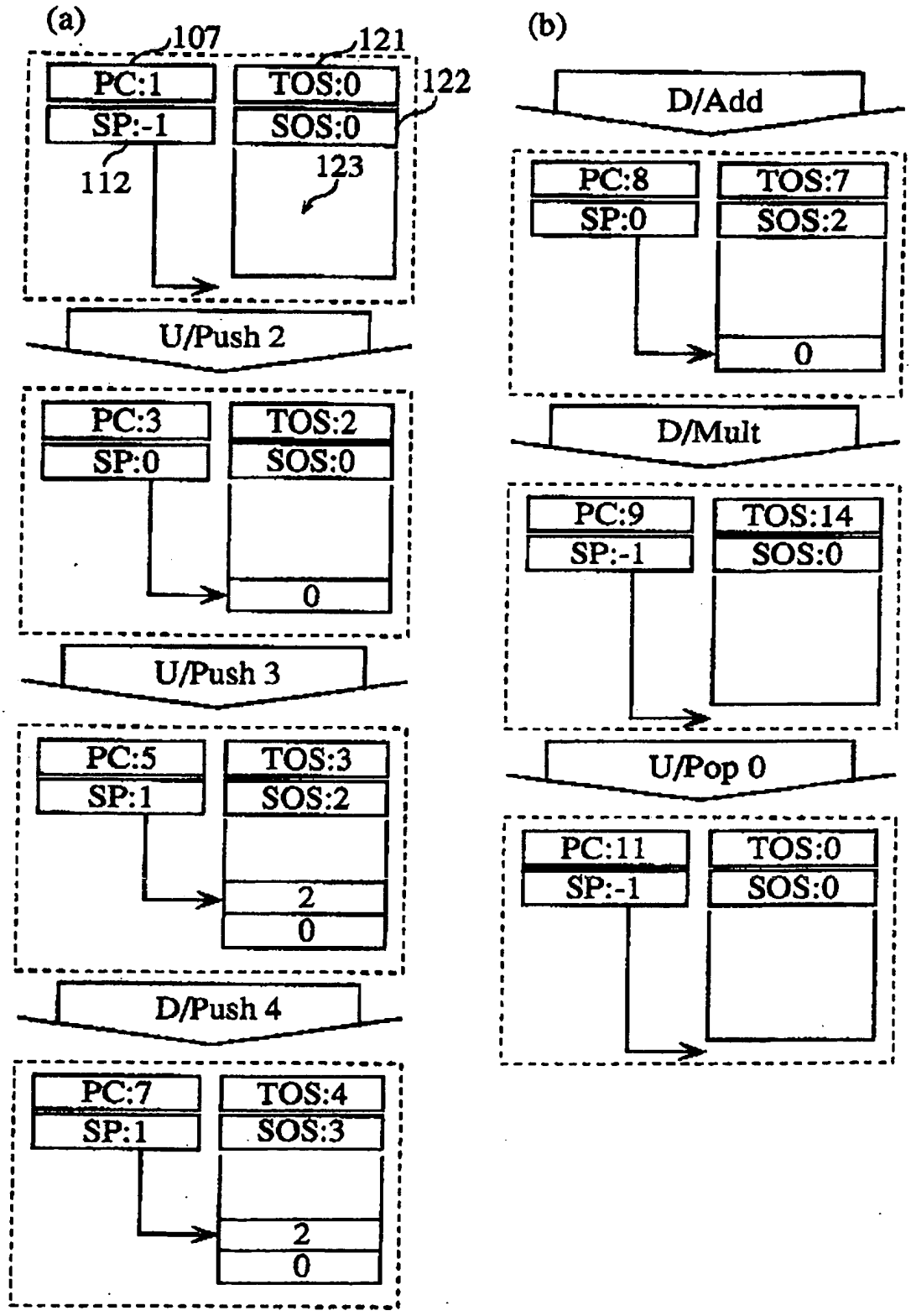
【図１２】


101
5101
テーブルのi番目のデコード
データを実行手段に出力する
5102
テーブルのi番目のオペランド数
をop Countに代入する
100
5103
i←0
5107
エラー処理
5104
Y
i≧op Count?
N
5105
仮想マシンコードを読み込んで実行
手段にオペランドとして出力する
5108
End
5106
i←i+1

【図１３】

1:＜上向きPushを処理するコードへのジャンプアドレス＞
2:オペランド"2"
3:＜上向きPushを処理するコードへのジャンプアドレス＞
4:オペランド"3"
5:＜下向きPushを処理するコードへのジャンプアドレス＞
6:オペランド"4"
7:＜下向きAddを処理するコードへのジャンプアドレス＞
8:＜下向きMultを処理するコードへのジャンプアドレス＞
9:＜上向きPopを処理するコードへのジャンプアドレス＞
10:オペランド"0"

【図１４】


(a)
107
121
PC:1
TOS:0
122
SP:-1
SOS:0
112
123
U/Push 2
PC:3
TOS:2
SP:0
SOS:0
0
U/Push 3
PC:5
TOS:3
SP:1
SOS:2
2
0
D/Push 4
PC:7
TOS:4
SP:1
SOS:3
2
0
(b)
D/Add
PC:8
TOS:7
SP:0
SOS:2
0
D/Mult
PC:9
TOS:14
SP:-1
SOS:0
U/Pop 0
PC:11
TOS:0
SP:-1
SOS:0

【図１５】

クロック	1	2	3	4	5	6	7	8	9	10	11
Load r1,[r2]	IF	RF	ALU	MEM	WB						
Load r4,[r3]		IF	RF	ALU	MEM	WB					
Inc r2			IF	RF	ALU	MEM	WB				
Dec r3				IF	RF	ALU	MEM	WB			
Jmp r1					IF	RF	ALU	MEM	WB		
						IF	x				
Mult r0,r0,r4							IF	RF	ALU	MEM	WB

【図１６】

クロック	1	2	3	4	5	6	7	8	9
Load r1,[r2]	IF	RF	ALU	MEM	WB				
Load r4,[r3]	IF	RF	ALU	MEM	WB				
Inc r2		IF	RF	ALU	MEM	WB			
Dec r3		IF	RF	ALU	MEM	WB			
Jmp r1			IF	RF	ALU	MEM	WB		
			IF	RF	x				
				IF	x				
				IF	x				
Mult r0,r0,r4					IF	RF	ALU	MEM	WB

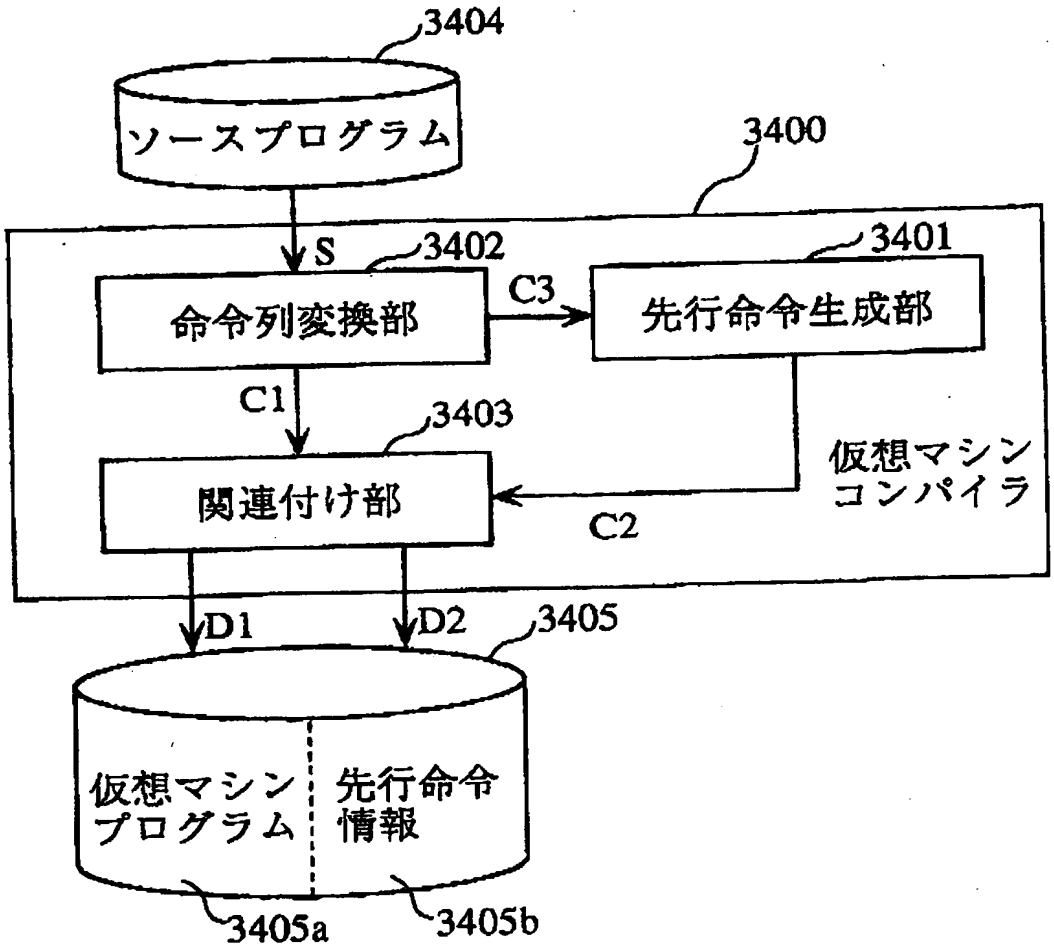
【図１７】

クロック	1	2	3	4	5	6	7	8	9	10	11
Load r1,[r2]	IF	RF	ALU	MEM	WB						
Load r4,[r3]		IF	RF	ALU	MEM	WB					
Inc r2			IF	RF	ALU	MEM	WB				
Dec r3				IF	RF	ALU	MEM	WB			
Jmp r1					IF	RF	ALU	MEM	WB		
						IF	x				
Mult r0,r0,r4							IF	RF	ALU	MEM	WB

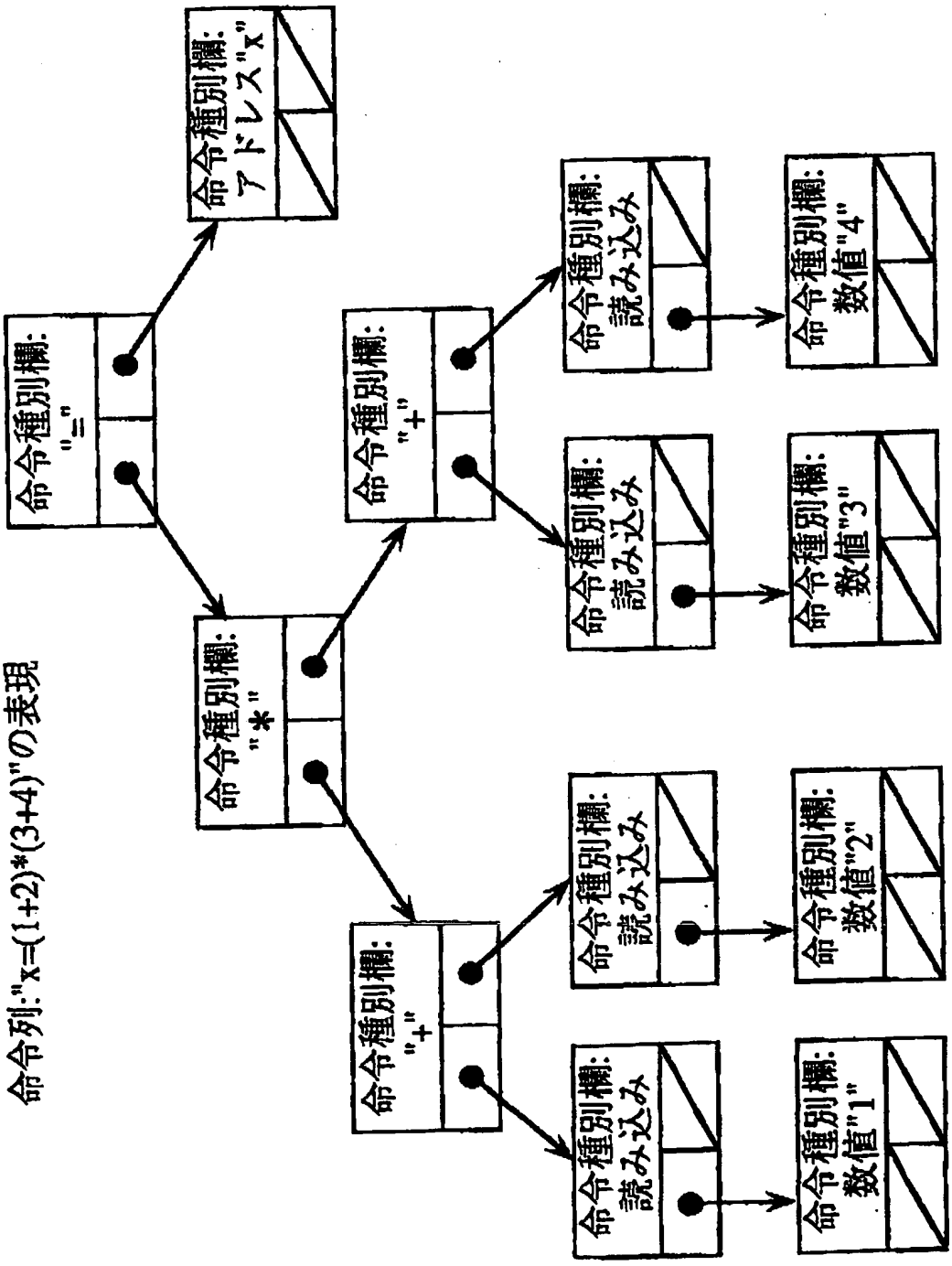
【図１８】

クロック	1	2	3	4	5	6	7	8	9	10
Load r1,[r2]	IF	RF	ALU	MEM	WB					
Load r4,[r3]	IF	RF	ALU	MEM	WB					
Inc r2		IF	RF	ALU	MEM	WB				
Dec r3		IF	RF	ALU	MEM	WB				
Jmp r1			IF	RF	・	ALU	MEM	WB		
			IF	RF	ALU	x				
				IF	RF	x				
				IF	RF	x				
					IF	x				
					IF	x				
Mult r0,r0,r4						IF	RF	ALU	MEM	WB

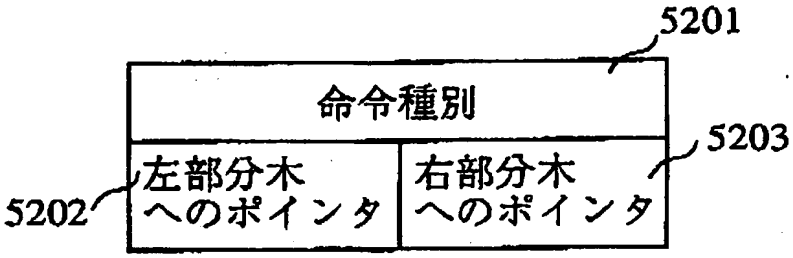
【図１９】


3404
ソースプログラム
3400
S
3402
命令列変換部
C3
3401
先行命令生成部
C1
3403
関連付け部
C2
仮想マシン
コンパイラ
D1
D2
3405
仮想マシン
プログラム
先行命令
情報
3405a
3405b

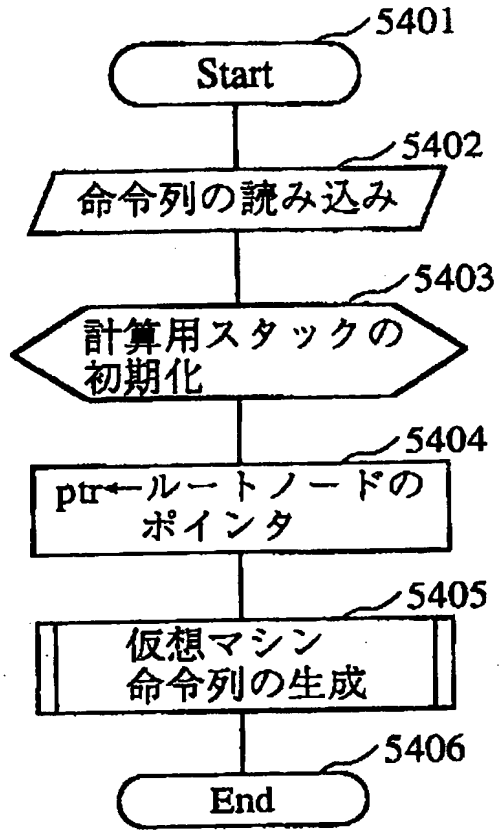
【図２０】


命令列:"x=(1+2)*(3+4)"の表現
命令種別欄:
"="
命令種別欄:
アドレス"x"
命令種別欄:
"*"
命令種別欄:
"+"
命令種別欄:
"+"
命令種別欄:
読み込み
命令種別欄:
読み込み
命令種別欄:
読み込み
命令種別欄:
読み込み
命令種別欄:
数値"1"
命令種別欄:
数値"2"
命令種別欄:
数値"3"
命令種別欄:
数値"4"

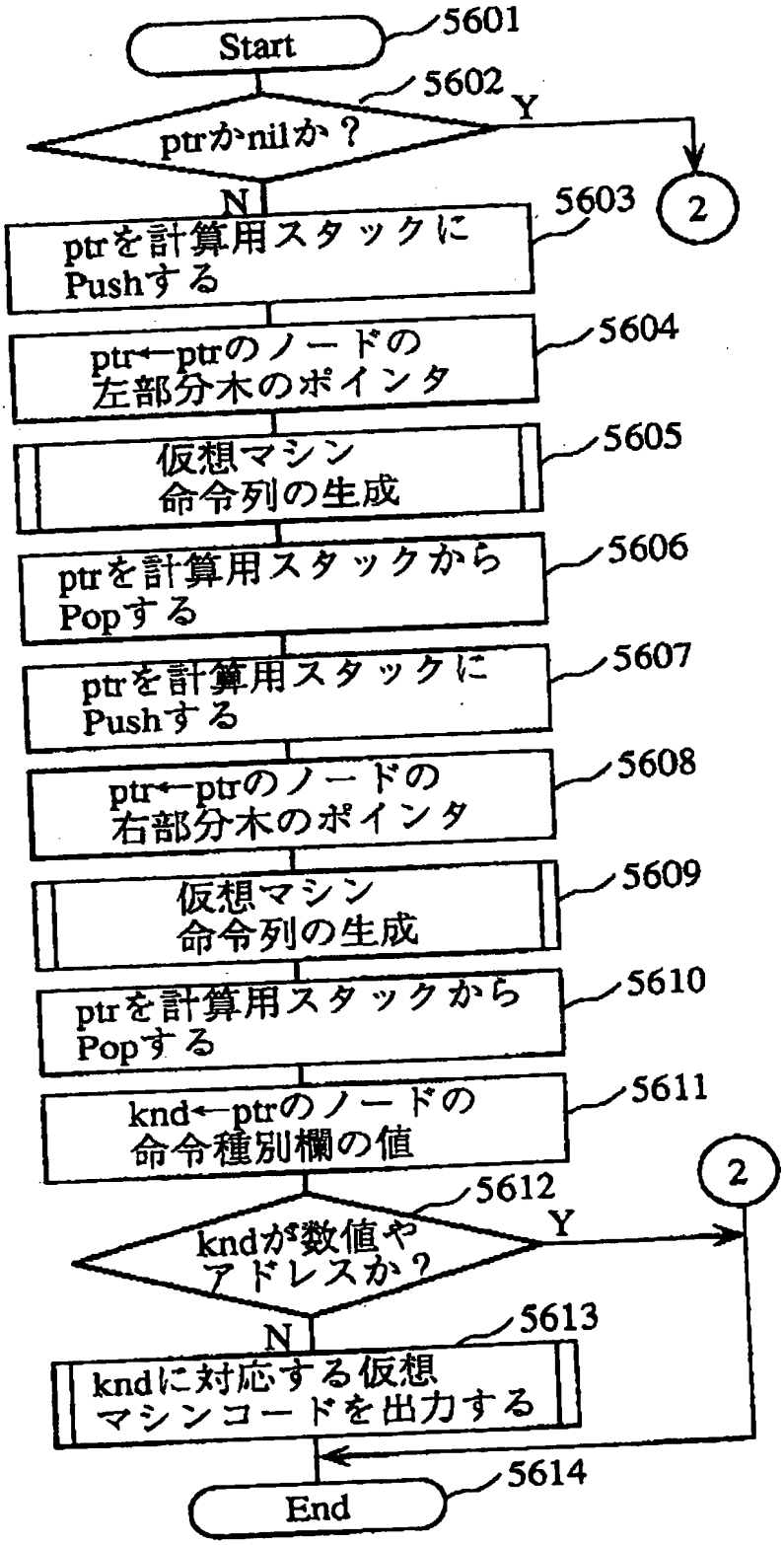
【図２１】



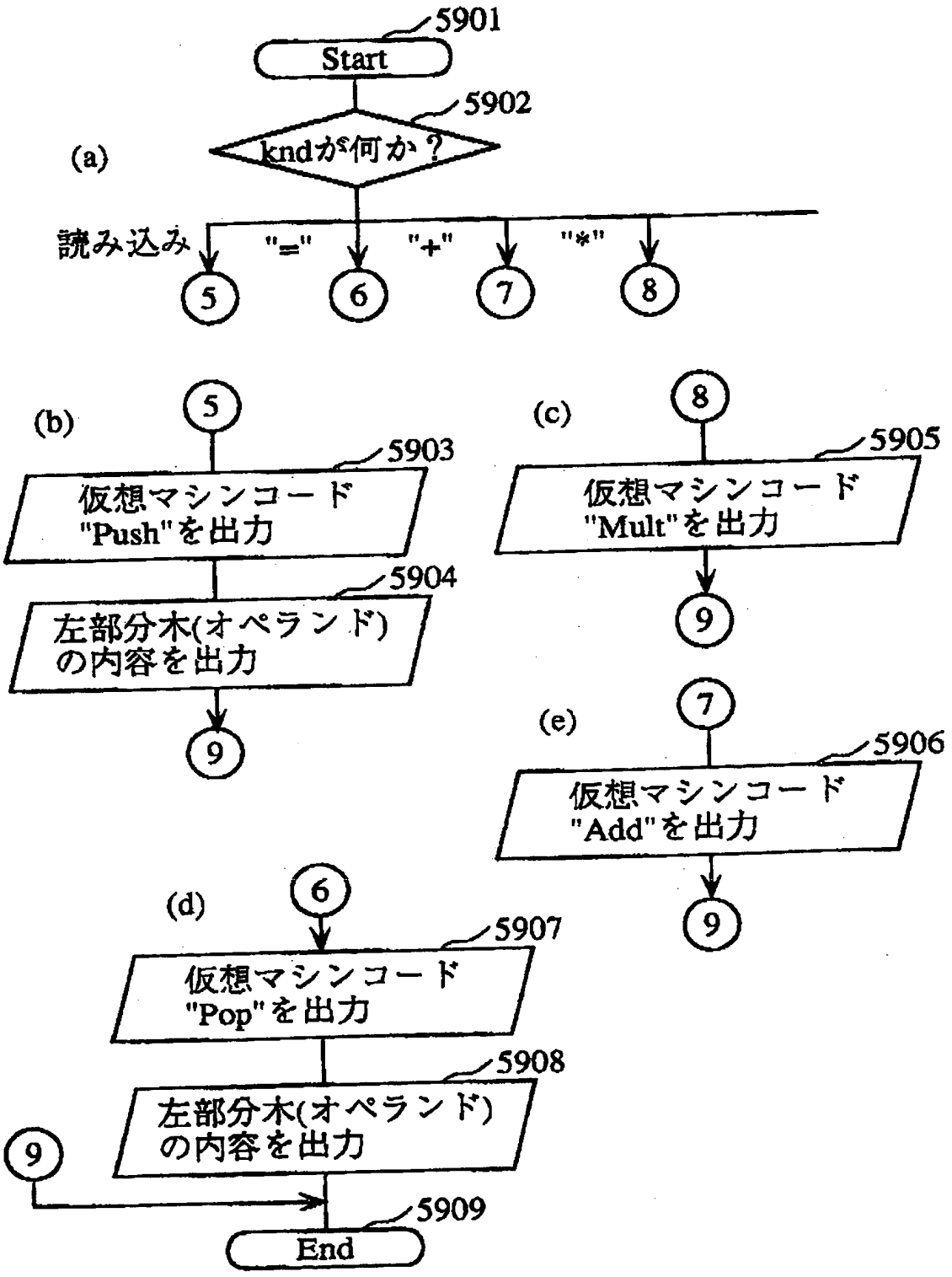
【図２２】


5401
Start
5402
命令列の読み込み
5403
計算用スタックの
初期化
5404
ptr←ルートノードの
ポインタ
5405
仮想マシン
命令列の生成
5406
End

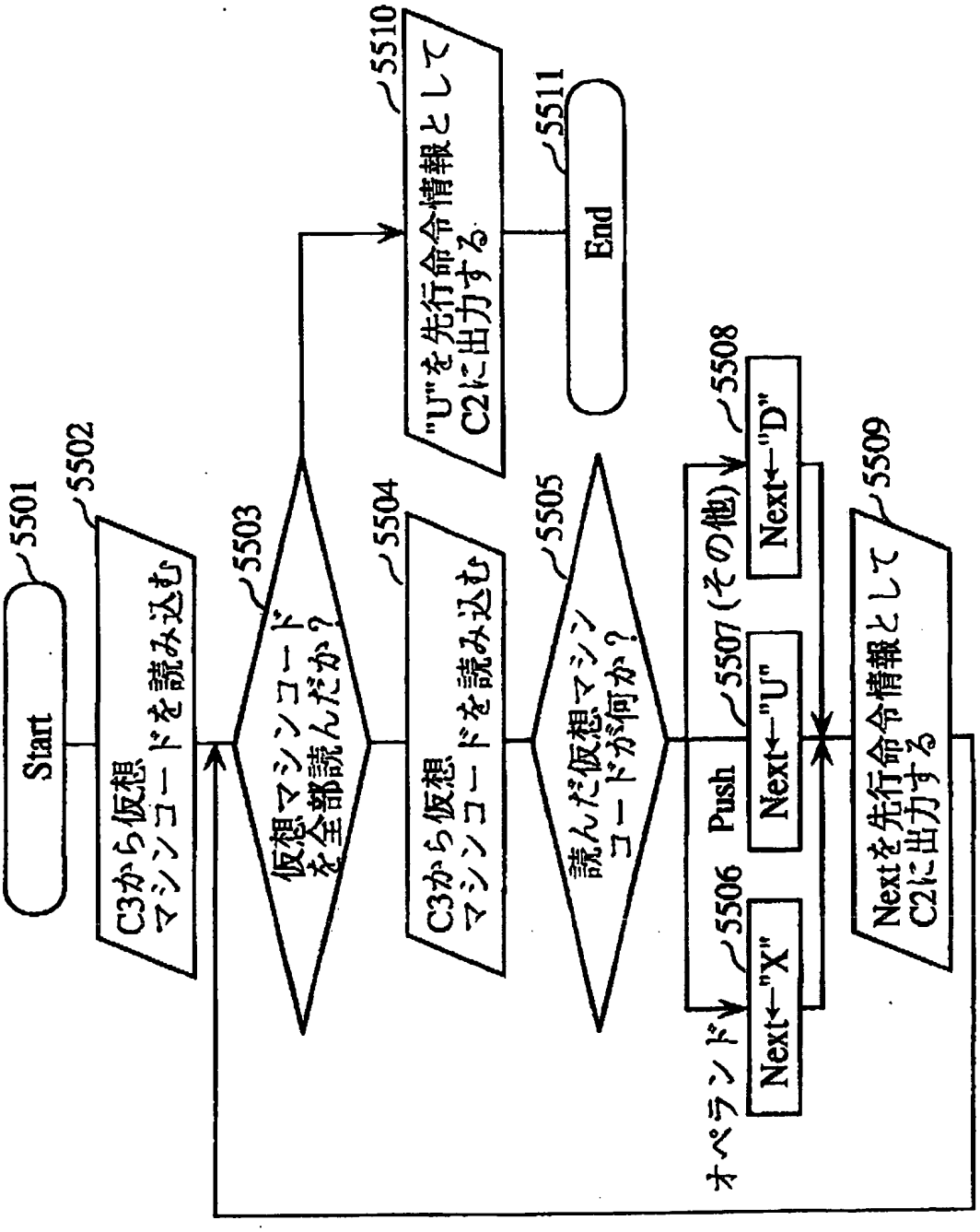
【図２３】


Start
5601
ptrかnilか？
5602
Y
N
2
ptrを計算用スタックに
Pushする
5603
ptr←ptrのノードの
左部分木のポインタ
5604
仮想マシン
命令列の生成
5605
ptrを計算用スタックから
Popする
5606
ptrを計算用スタックに
Pushする
5607
ptr←ptrのノードの
右部分木のポインタ
5608
仮想マシン
命令列の生成
5609
ptrを計算用スタックから
Popする
5610
knd←ptrのノードの
命令種別欄の値
5611
2
kndが数値や
アドレスか？
5612
Y
N
5613
kndに対応する仮想
マシンコードを出力する
End
5614

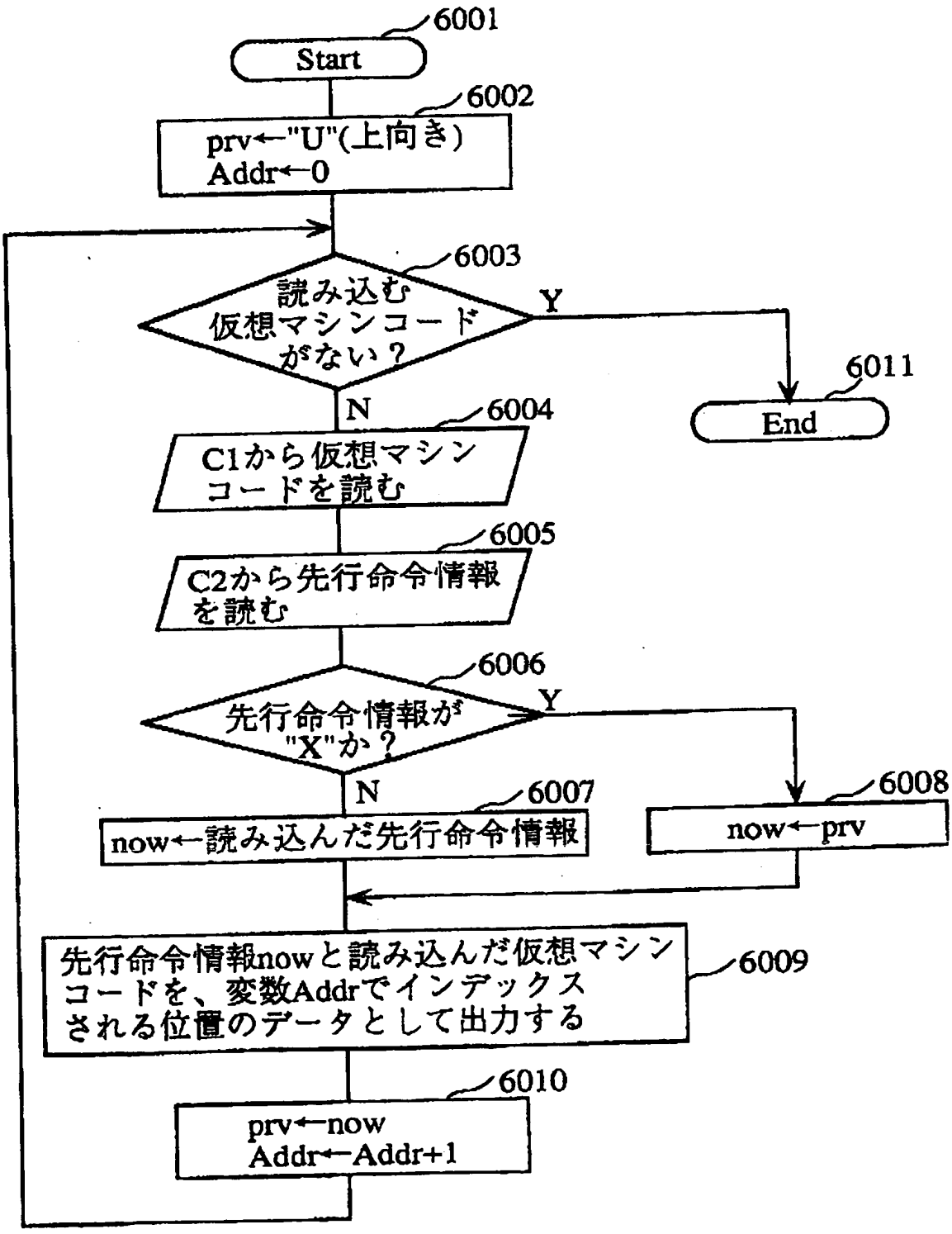
【図２４】


5901
Start
5902
(a)
kndが何か？
読み込み
"="
"+"
"*"
5
6
7
8
(b)
5
5903
仮想マシンコード
"Push"を出力
5904
左部分木(オペランド)
の内容を出力
9
(c)
8
5905
仮想マシンコード
"Mult"を出力
9
(e)
7
5906
仮想マシンコード
"Add"を出力
9
(d)
6
5907
仮想マシンコード
"Pop"を出力
5908
左部分木(オペランド)
の内容を出力
9
5909
End

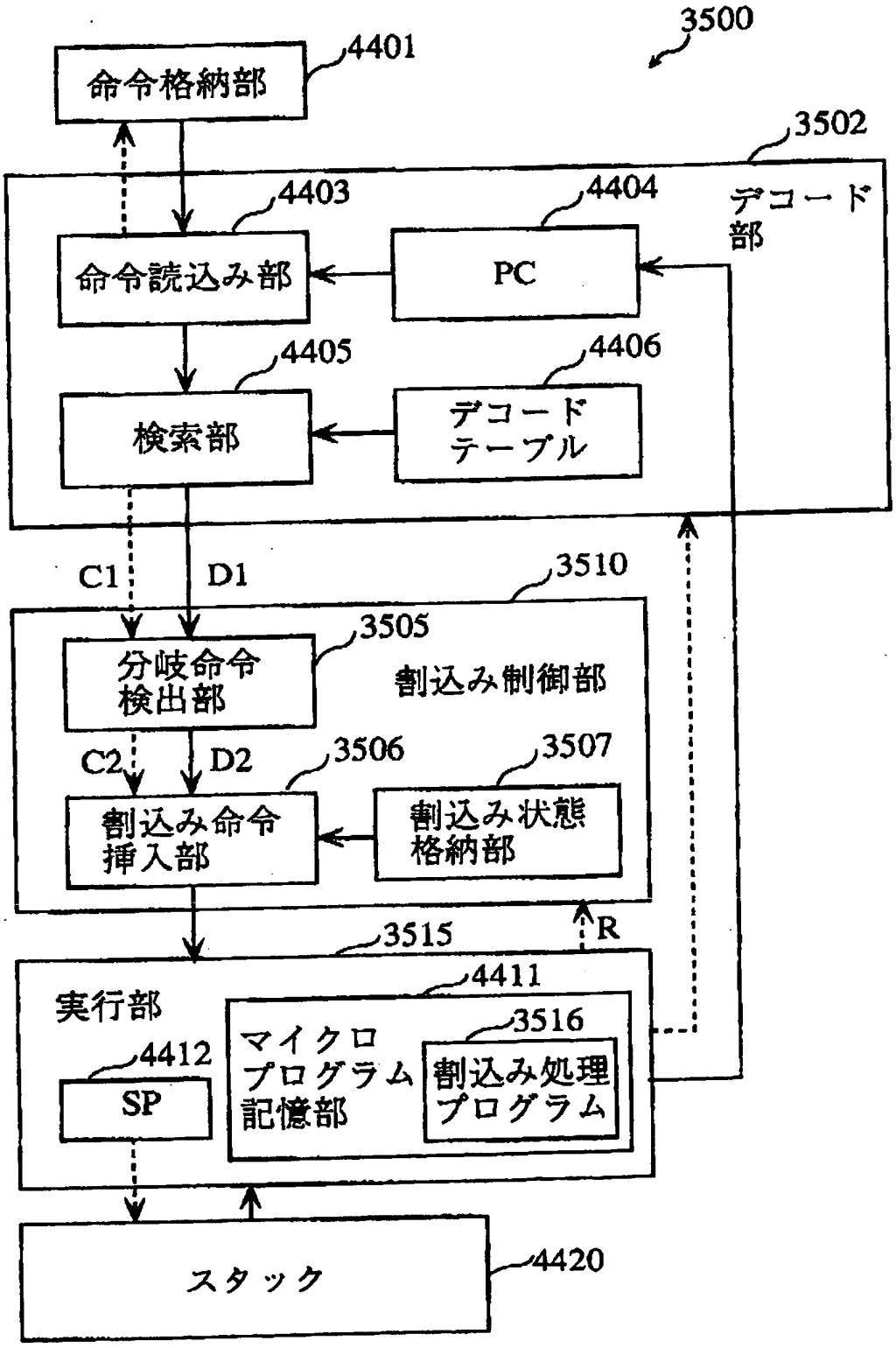
【図２５】


Start
5501
C3から仮想マシンコードを読み込む
5502
仮想マシンコードを全部読んだか？
5503
C3から仮想マシンコードを読み込む
5504
読んだ仮想マシンコードが何か？
5505
オペランド
Push
5506
Next←"X"
5507
Next←"U"
(その他)
5508
Next←"D"
Nextを先行命令情報としてC2に出力する
5509
"U"を先行命令情報としてC2に出力する
5510
End
5511

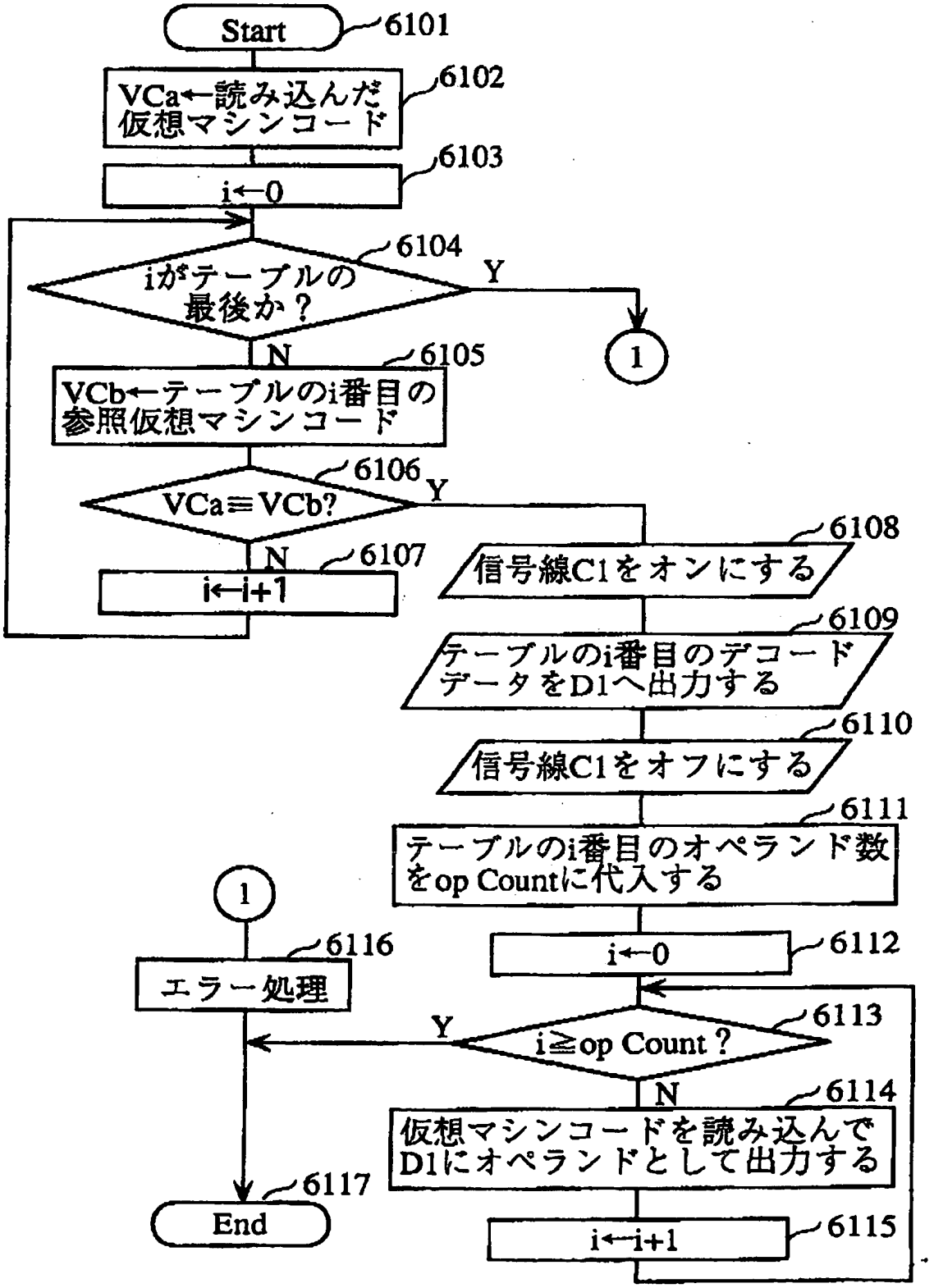
【図２６】


6001
Start
6002
prv←"U"(上向き)
Addr←0
6003
読み込む
仮想マシンコード
がない？
Y
N
6011
End
6004
C1から仮想マシン
コードを読む
6005
C2から先行命令情報
を読む
6006
先行命令情報が
"X"か？
Y
N
6007
now←読み込んだ先行命令情報
6008
now←prv
先行命令情報nowと読み込んだ仮想マシン
コードを、変数Addrでインデックス
される位置のデータとして出力する
6009
6010
prv←now
Addr←Addr+1

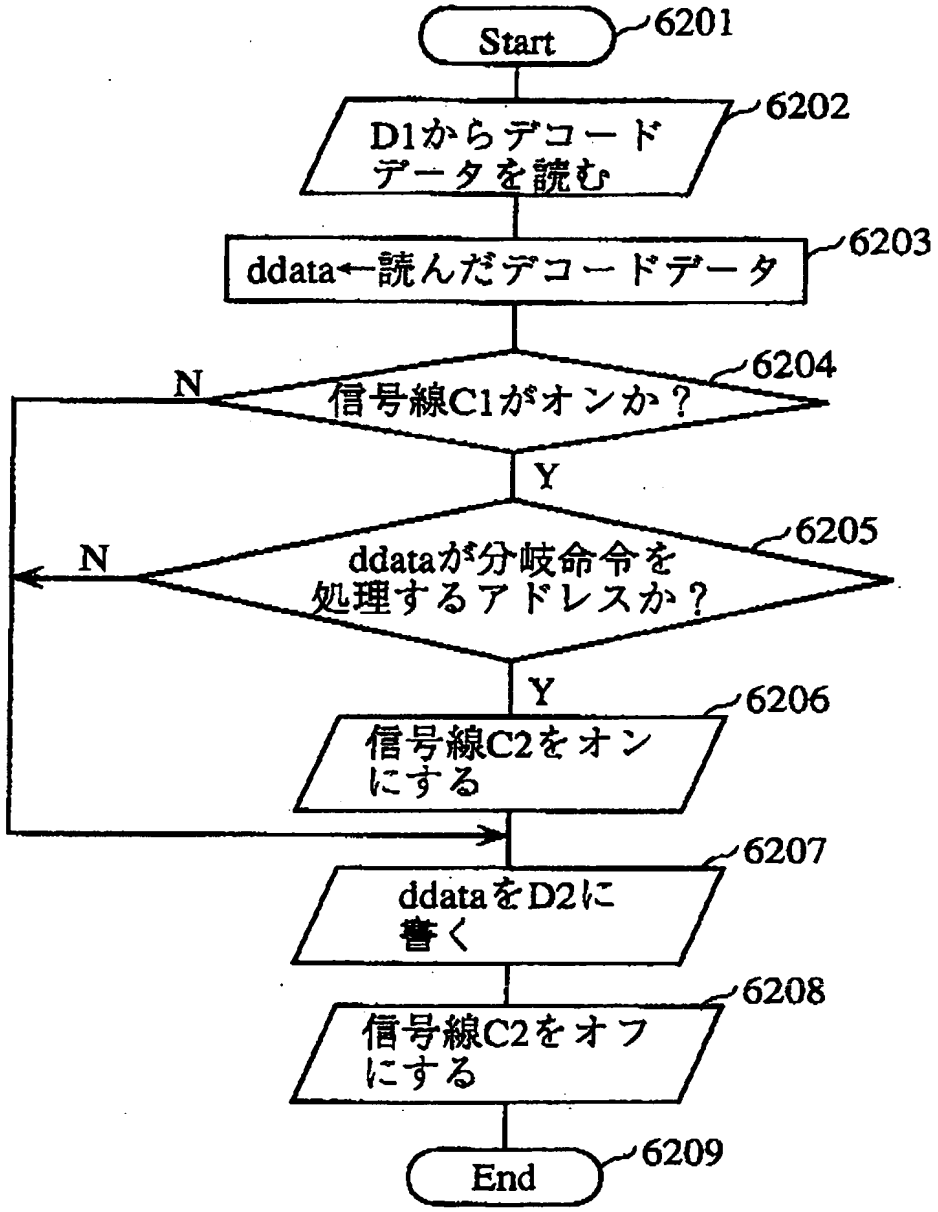
【図２７】


3500
4401
命令格納部
3502
4403
4404
デコード部
命令読込み部
PC
4405
4406
検索部
デコードテーブル
C1
D1
3510
3505
分岐命令検出部
割込み制御部
C2
D2
3506
3507
割込み命令挿入部
割込み状態格納部
R
3515
実行部
4411
マイクロプログラム記憶部
3516
割込み処理プログラム
4412
SP
スタック
4420

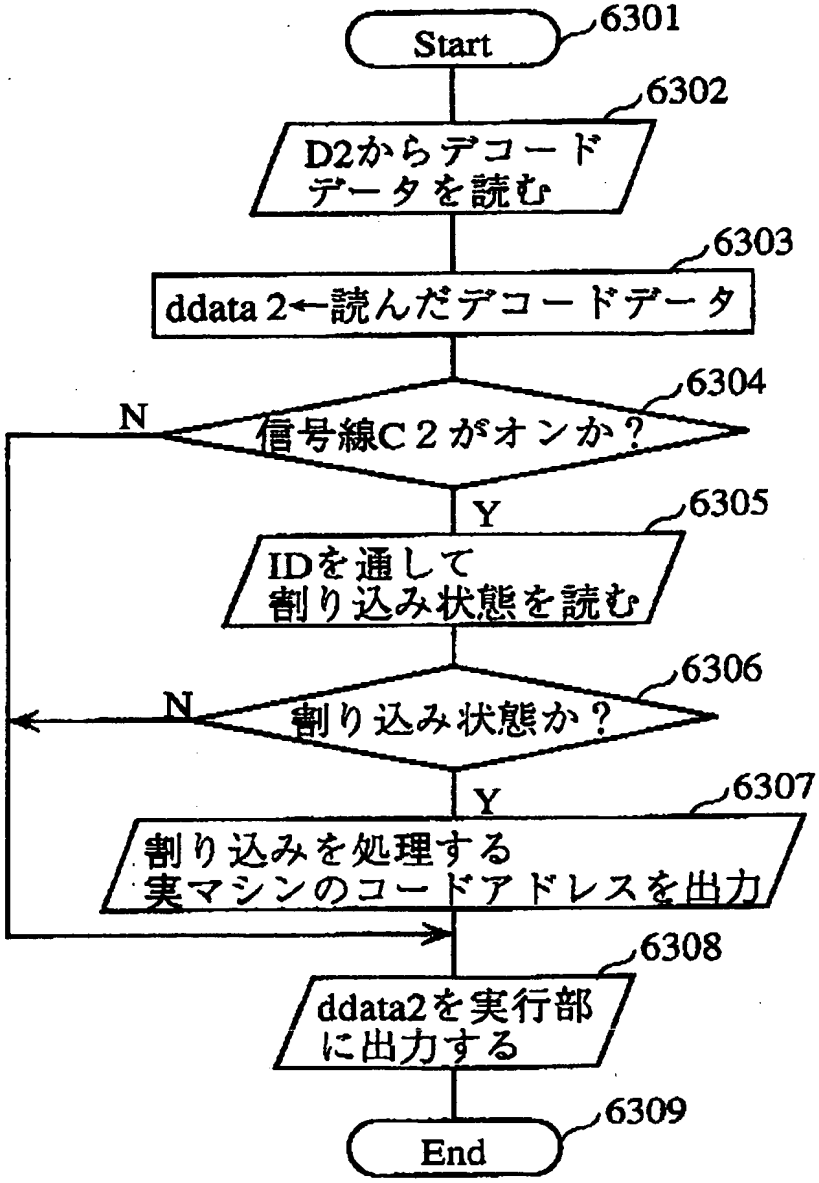
【図２８】


Start
6101
VCa←読み込んだ仮想マシンコード
6102
i←0
6103
iがテーブルの最後か？
6104
Y
N
1
VCb←テーブルのi番目の参照仮想マシンコード
6105
VCa≡VCb?
6106
Y
N
i←i+1
6107
信号線C1をオンにする
6108
テーブルのi番目のデコードデータをD1へ出力する
6109
信号線C1をオフにする
6110
テーブルのi番目のオペランド数をop Countに代入する
6111
i←0
6112
i≧op Count？
6113
Y
N
仮想マシンコードを読み込んでD1にオペランドとして出力する
6114
i←i+1
6115
1
エラー処理
6116
End
6117

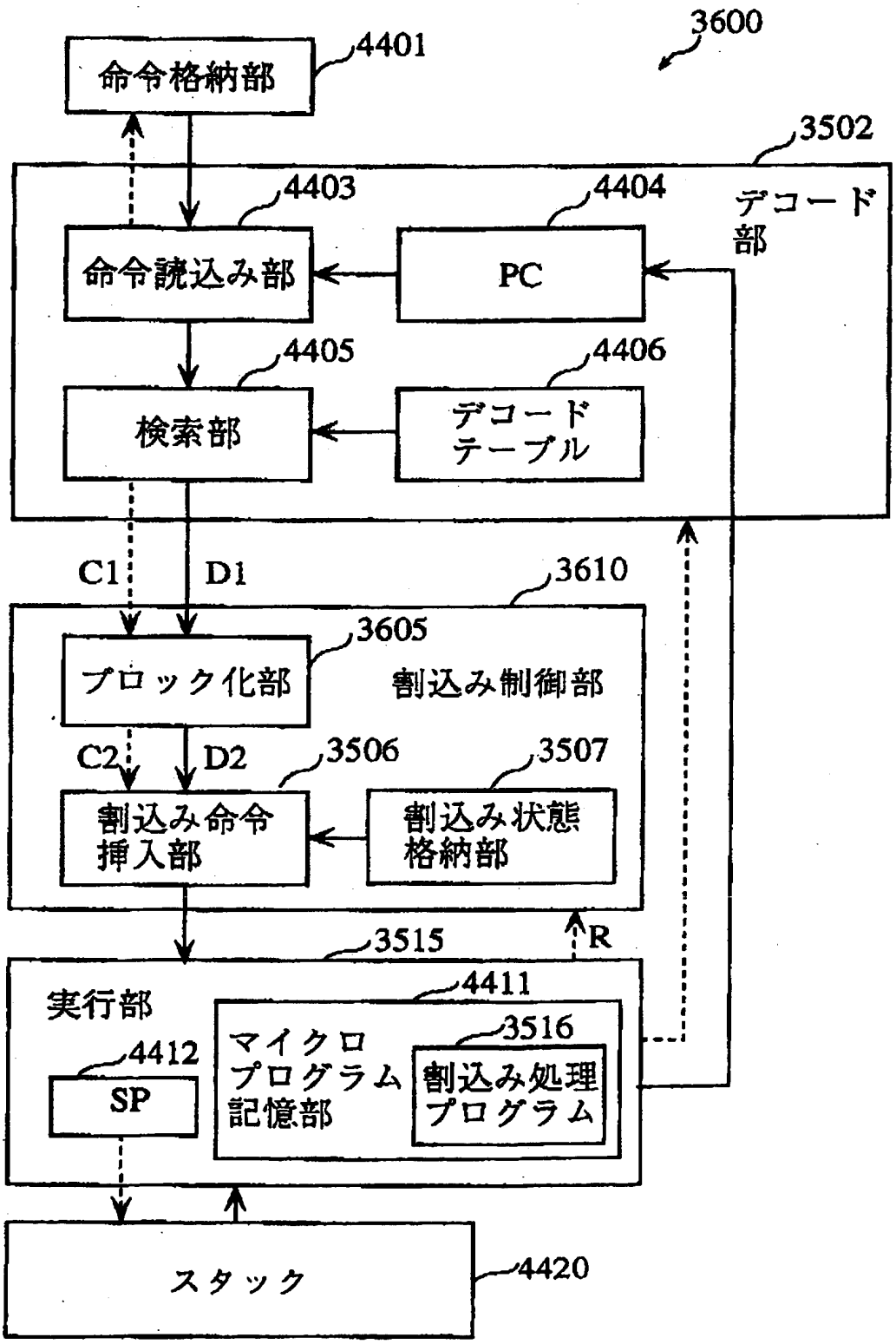
【図２９】


Start
6201
D1からデコード
データを読む
6202
ddata←読んだデコードデータ
6203
信号線C1がオンか？
6204
N
Y
ddataが分岐命令を
処理するアドレスか？
6205
N
Y
信号線C2をオン
にする
6206
ddataをD2に
書く
6207
信号線C2をオフ
にする
6208
End
6209

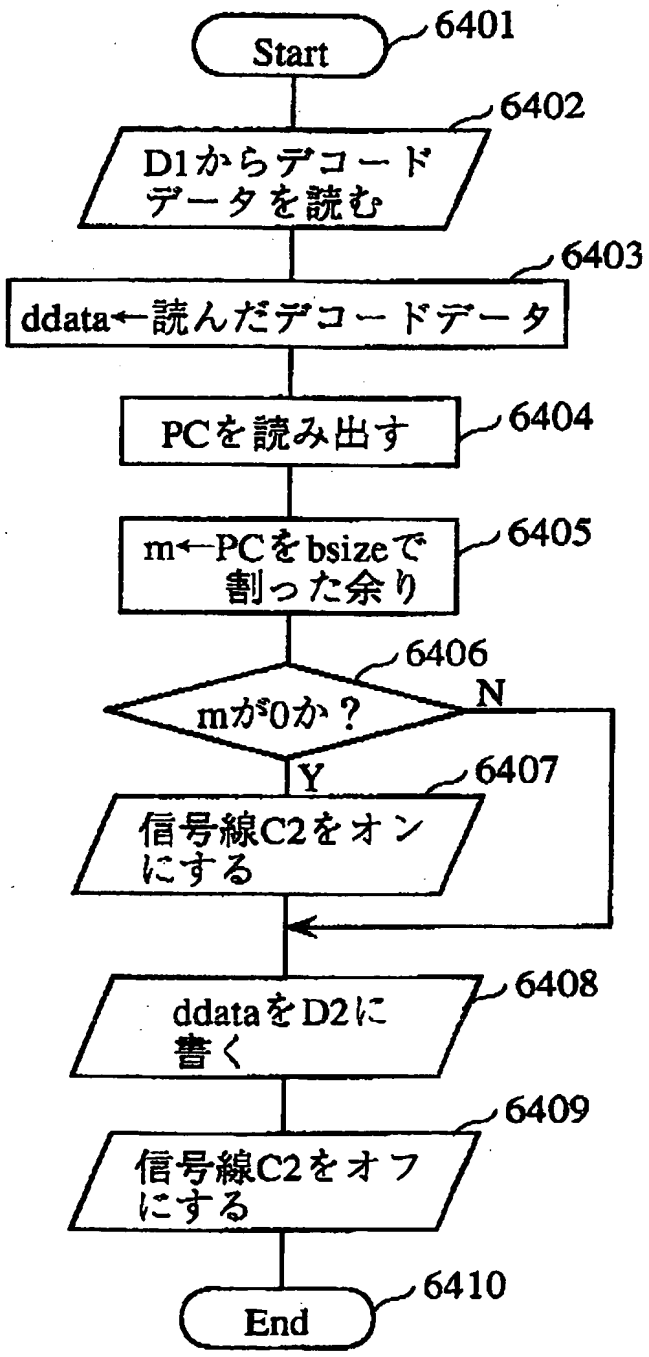
【図３０】


Start
6301
D2からデコード
データを読む
6302
ddata 2←読んだデコードデータ
6303
信号線C２がオンか？
6304
N
Y
IDを通して
割り込み状態を読む
6305
割り込み状態か？
6306
N
Y
割り込みを処理する
実マシンのコードアドレスを出力
6307
ddata2を実行部
に出力する
6308
End
6309

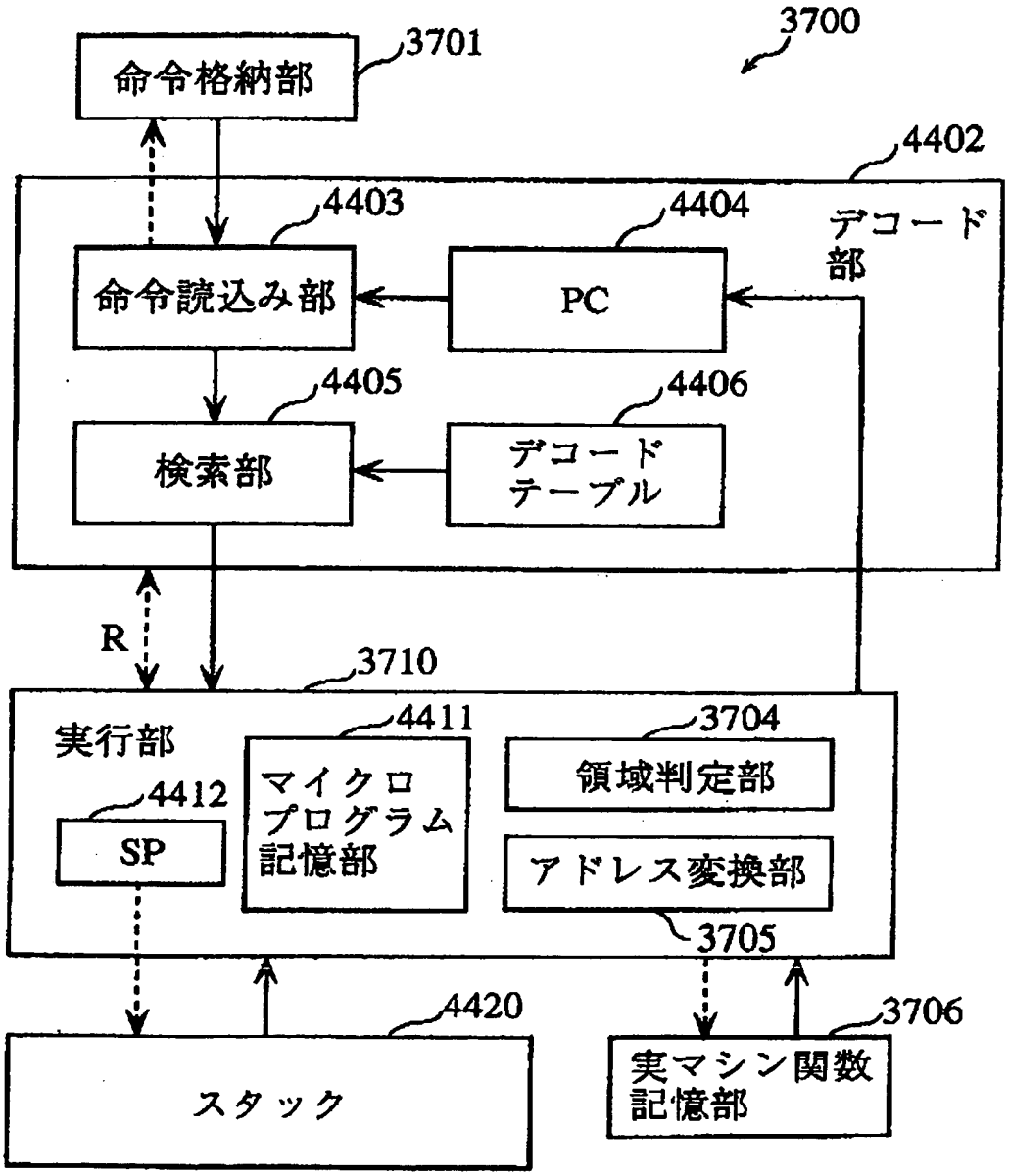
【図３１】


3600
命令格納部
4401
3502
4403
4404
デコード部
命令読込み部
PC
4405
4406
検索部
デコードテーブル
C1
D1
3610
3605
ブロック化部
割込み制御部
C2
D2
3506
3507
割込み命令挿入部
割込み状態格納部
3515
R
実行部
4411
マイクロプログラム記憶部
3516
割込み処理プログラム
4412
SP
スタック
4420

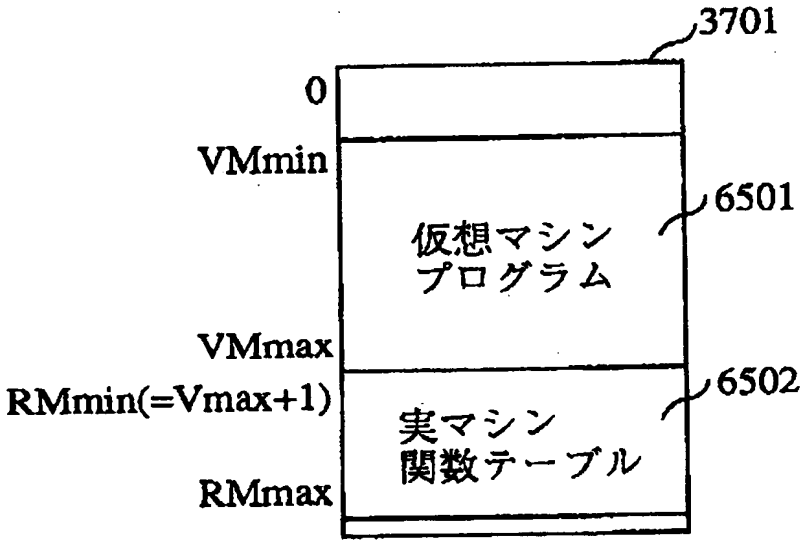
【図３２】


Start
6401
D1からデコード
データを読む
6402
ddata←読んだデコードデータ
6403
PCを読み出す
6404
m←PCをbsizeで
割った余り
6405
mが0か？
6406
N
Y
信号線C2をオン
にする
6407
ddataをD2に
書く
6408
信号線C2をオフ
にする
6409
End
6410

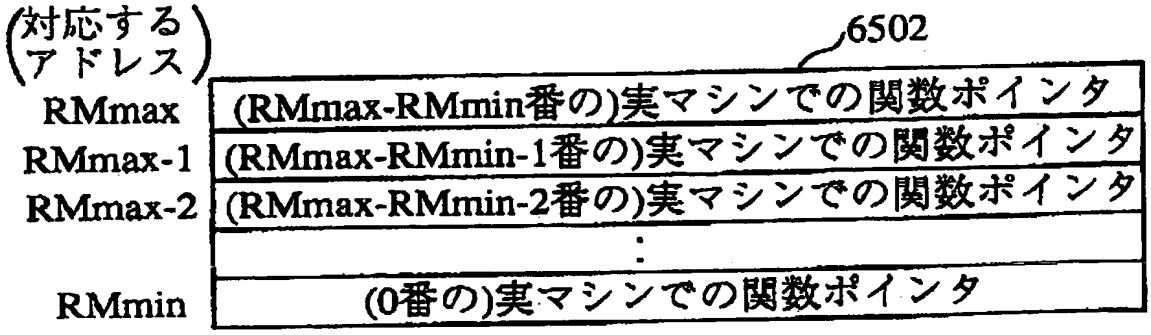
【図３３】


3700
命令格納部
3701
4402
デコード部
4403
命令読込み部
4404
PC
4405
検索部
4406
デコードテーブル
R
3710
実行部
4411
マイクロプログラム記憶部
4412
SP
3704
領域判定部
アドレス変換部
3705
4420
スタック
3706
実マシン関数記憶部

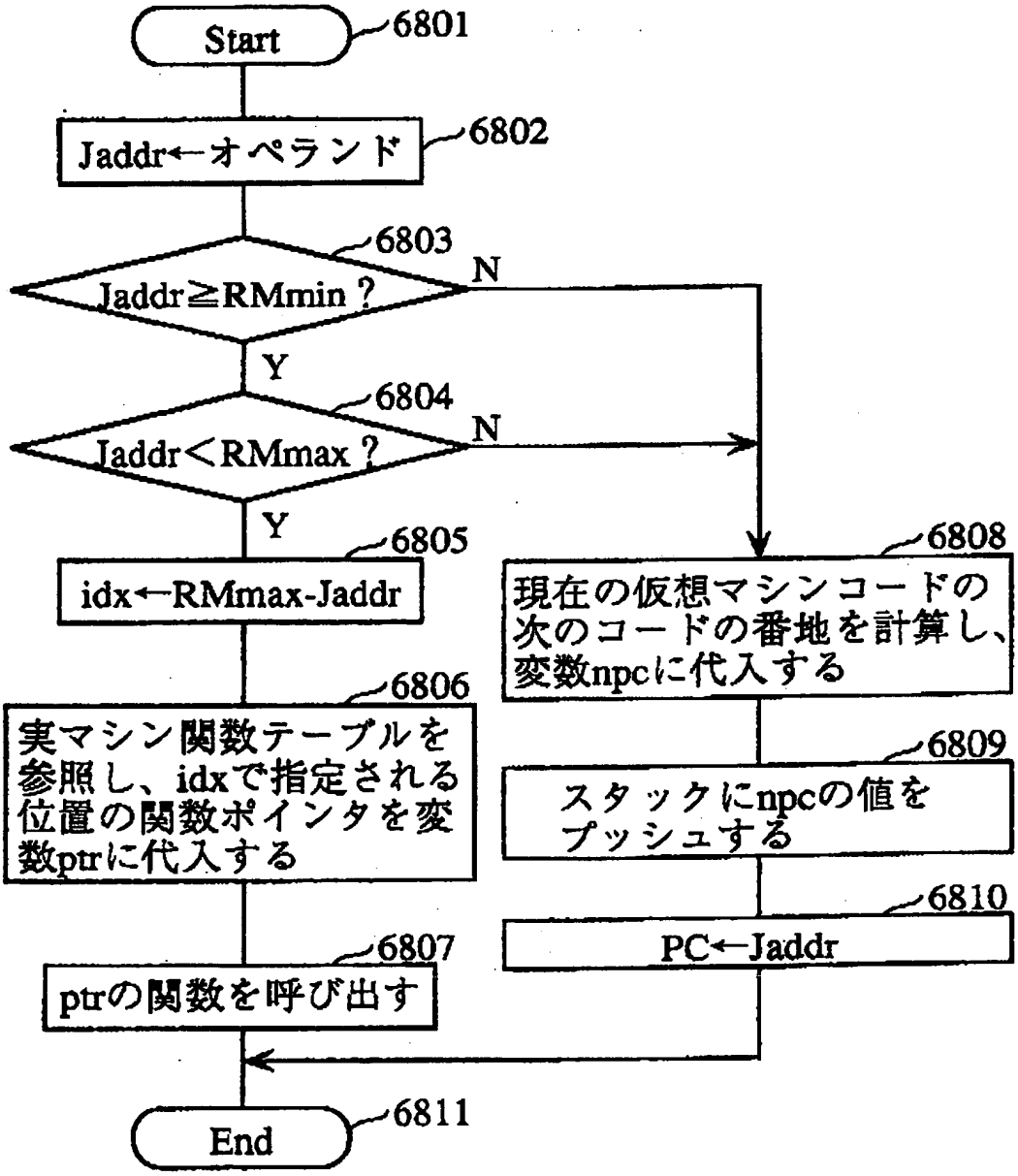
【図３４】


3701
0
VMmin
6501
仮想マシン
プログラム
VMmax
RMmin(=Vmax+1)
6502
実マシン
関数テーブル
RMmax

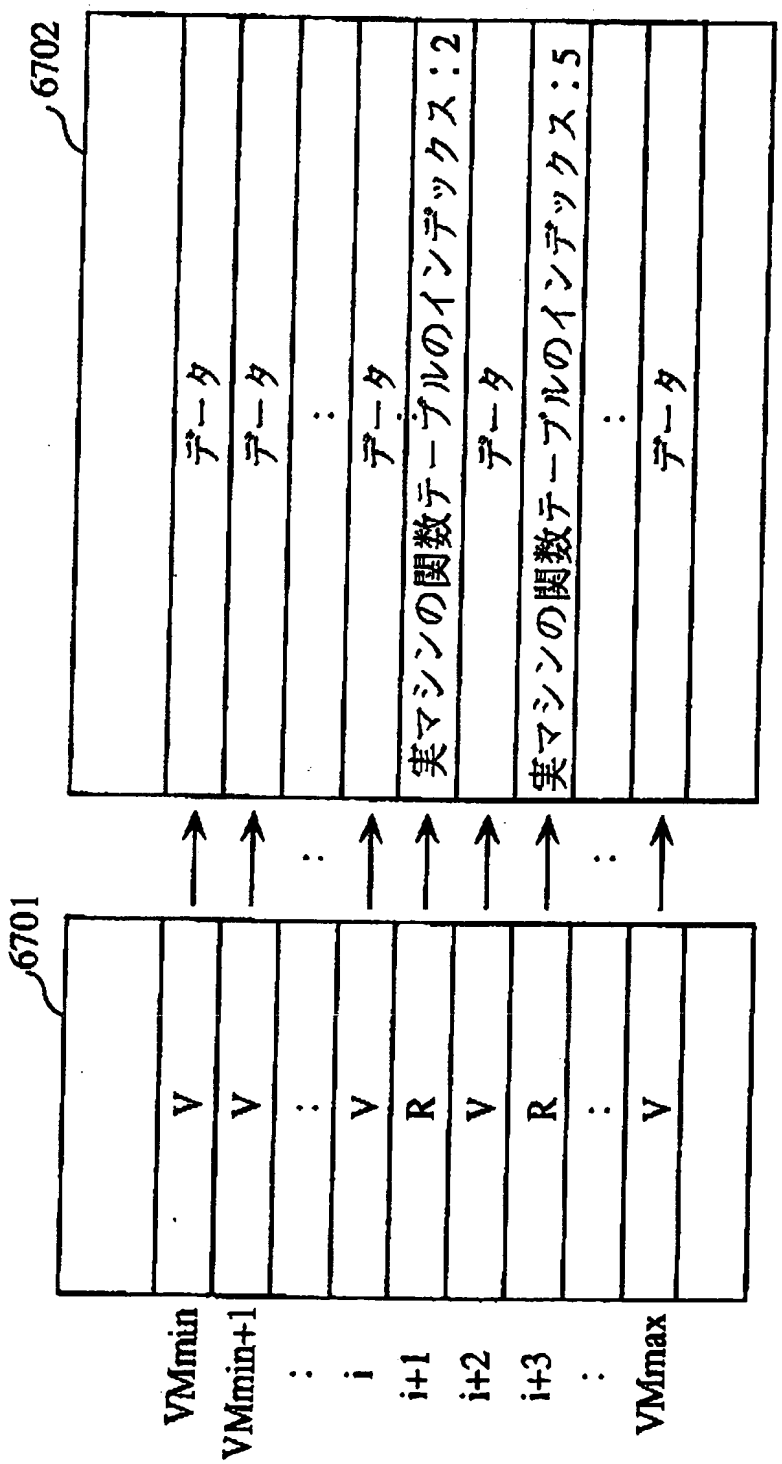
【図３５】


(対応する
アドレス)
6502
RMmax
(RMmax-RMmin番の)実マシンでの関数ポインタ
RMmax-1
(RMmax-RMmin-1番の)実マシンでの関数ポインタ
RMmax-2
(RMmax-RMmin-2番の)実マシンでの関数ポインタ
RMmin
(0番の)実マシンでの関数ポインタ

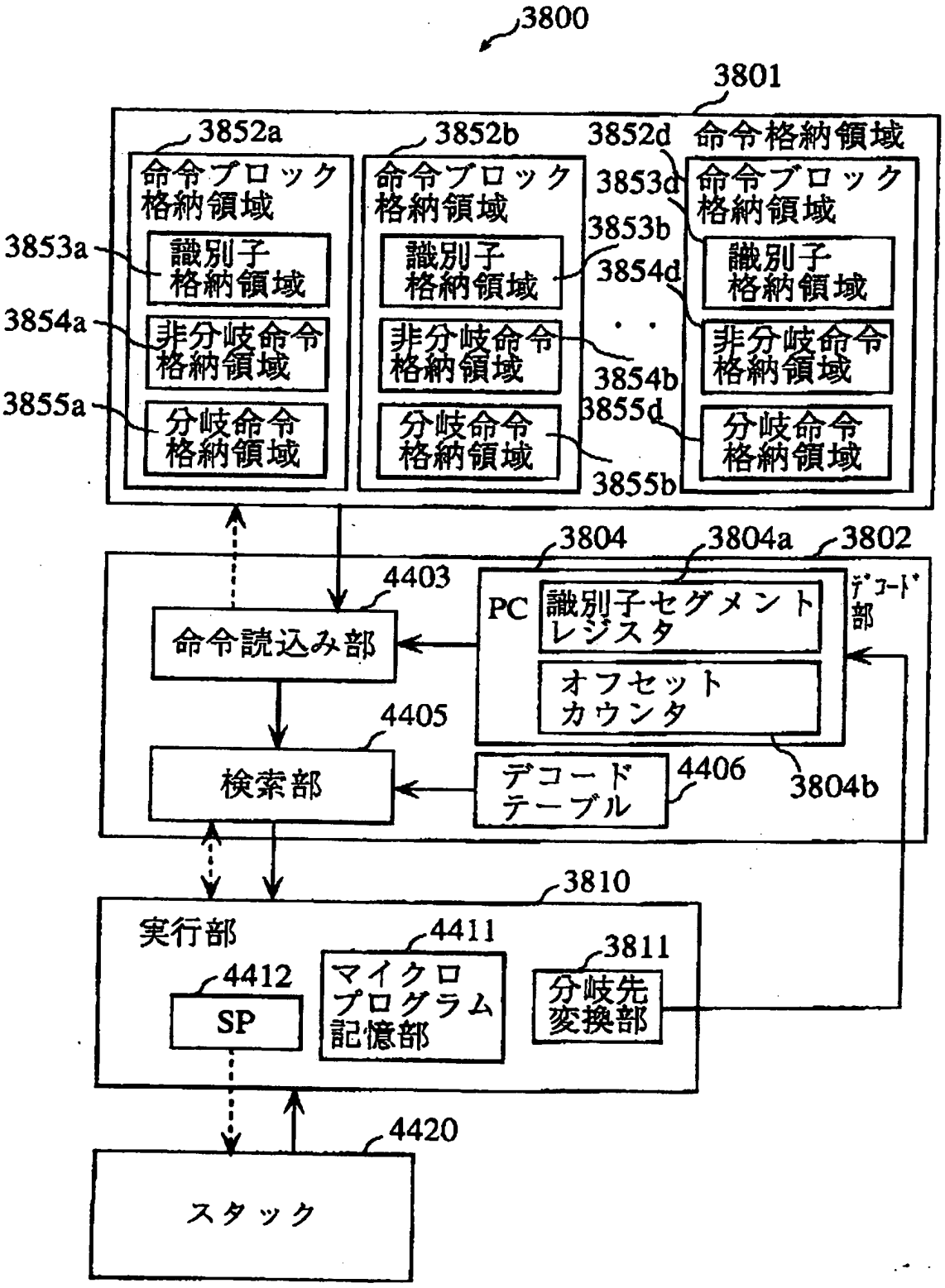
【図３６】


Start
6801
Jaddr←オペランド
6802
6803
Jaddr≧RMmin ?
N
Y
6804
Jaddr＜RMmax ?
N
Y
6805
idx←RMmax-Jaddr
6806
実マシン関数テーブルを参照し、idxで指定される位置の関数ポインタを変数ptrに代入する
6807
ptrの関数を呼び出す
6808
現在の仮想マシンコードの次のコードの番地を計算し、変数npcに代入する
6809
スタックにnpcの値をプッシュする
6810
PC←Jaddr
End
6811

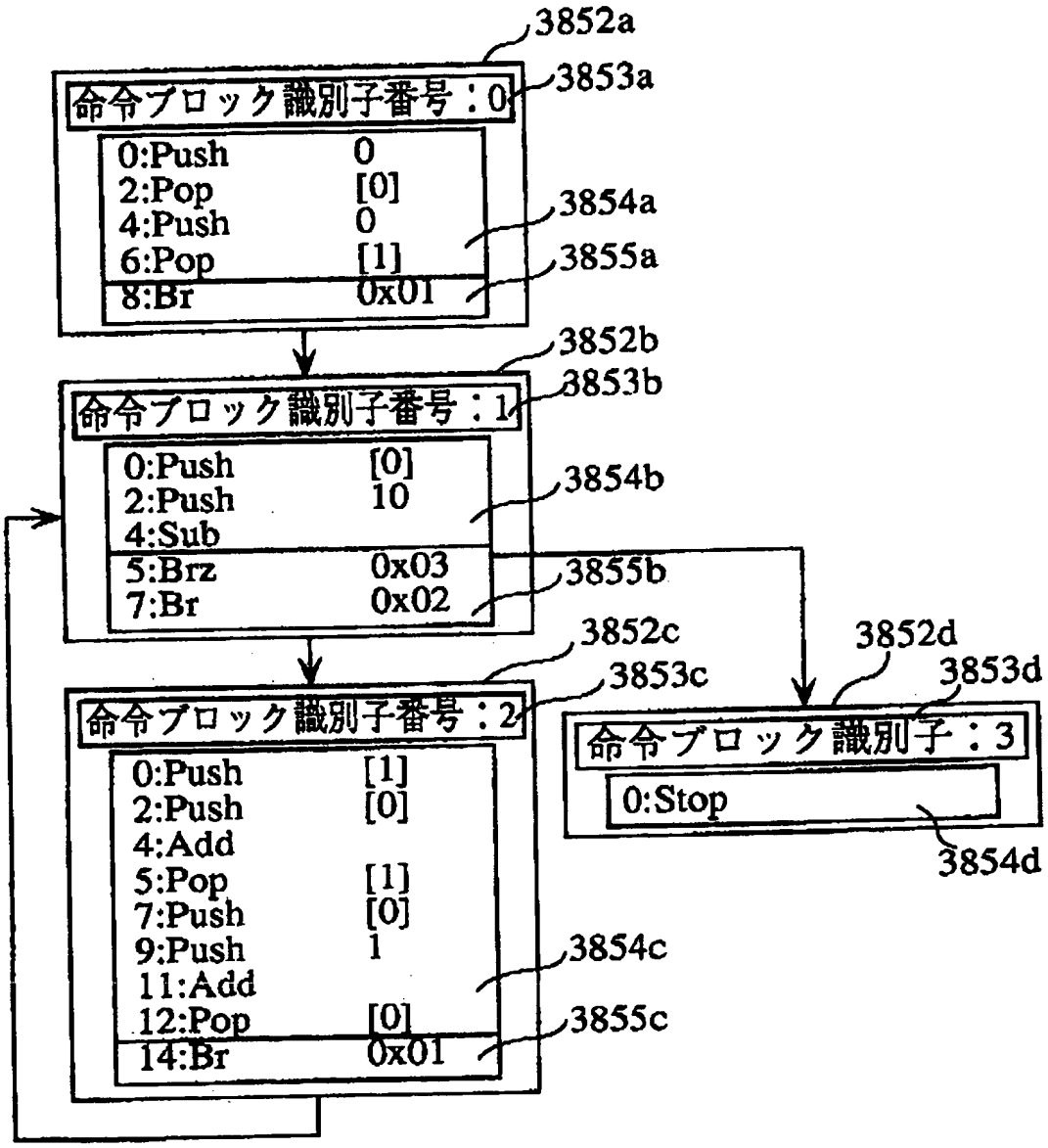
【図３７】


6702
データ
データ
：
データ
実マシンの関数テーブルのインデックス：2
データ
実マシンの関数テーブルのインデックス：5
：
データ
6701
V
V
：
V
R
V
R
：
V
VMmin
VMmin+1
：
i
i+1
i+2
i+3
：
VMmax

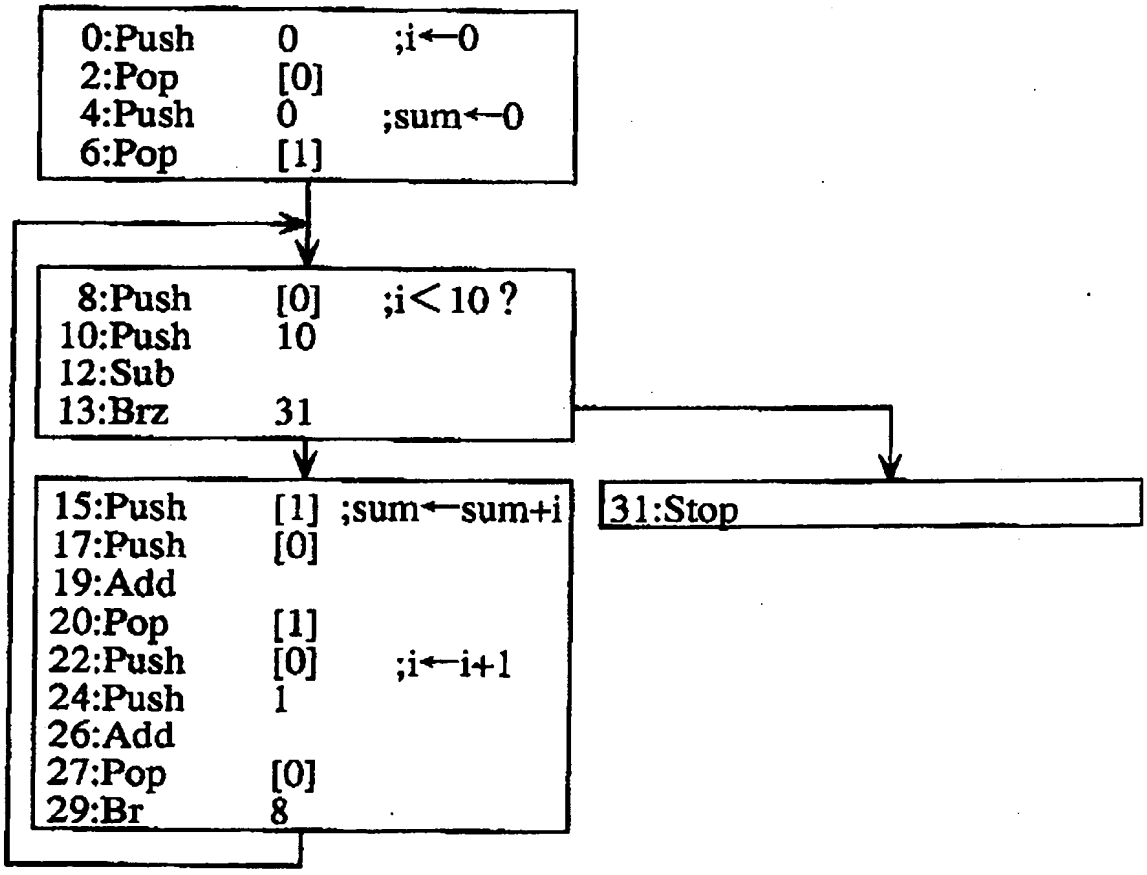
【図３８】


3800
3801
命令格納領域
3852a
命令ブロック格納領域
3852b
命令ブロック格納領域
3852d
命令ブロック格納領域
3853a
識別子格納領域
3853b
識別子格納領域
3853d
識別子格納領域
3854a
非分岐命令格納領域
3854b
非分岐命令格納領域
3854d
非分岐命令格納領域
3855a
分岐命令格納領域
3855b
分岐命令格納領域
3855d
分岐命令格納領域
3804
3804a
3802
デコード部
PC
識別子セグメントレジスタ
オフセットカウンタ
3804b
4403
命令読込み部
4405
検索部
デコードテーブル
4406
3810
実行部
4411
マイクロプログラム記憶部
4412
SP
3811
分岐先変換部
4420
スタック

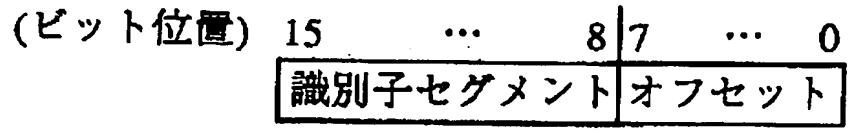
【図３９】


3852a
命令ブロック識別子番号：0
3853a
0:Push 0
2:Pop [0]
4:Push 0
6:Pop [1]
8:Br 0x01
3854a
3855a
3852b
命令ブロック識別子番号：1
3853b
0:Push [0]
2:Push 10
4:Sub
5:Brz 0x03
7:Br 0x02
3854b
3855b
3852c
命令ブロック識別子番号：2
3853c
0:Push [1]
2:Push [0]
4:Add
5:Pop [1]
7:Push [0]
9:Push 1
11:Add
12:Pop [0]
14:Br 0x01
3854c
3855c
3852d
命令ブロック識別子：3
3853d
0:Stop
3854d

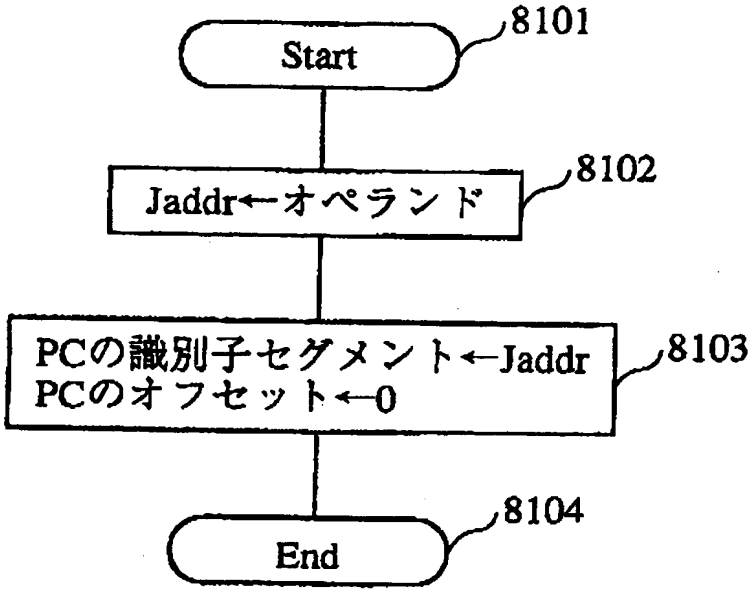
【図４０】


0:Push 0 ;i←0
2:Pop [0]
4:Push 0 ;sum←0
6:Pop [1]
8:Push [0] ;i＜10？
10:Push 10
12:Sub
13:Brz 31
15:Push [1] ;sum←sum+i
17:Push [0]
19:Add
20:Pop [1]
22:Push [0] ;i←i+1
24:Push 1
26:Add
27:Pop [0]
29:Br 8
31:Stop

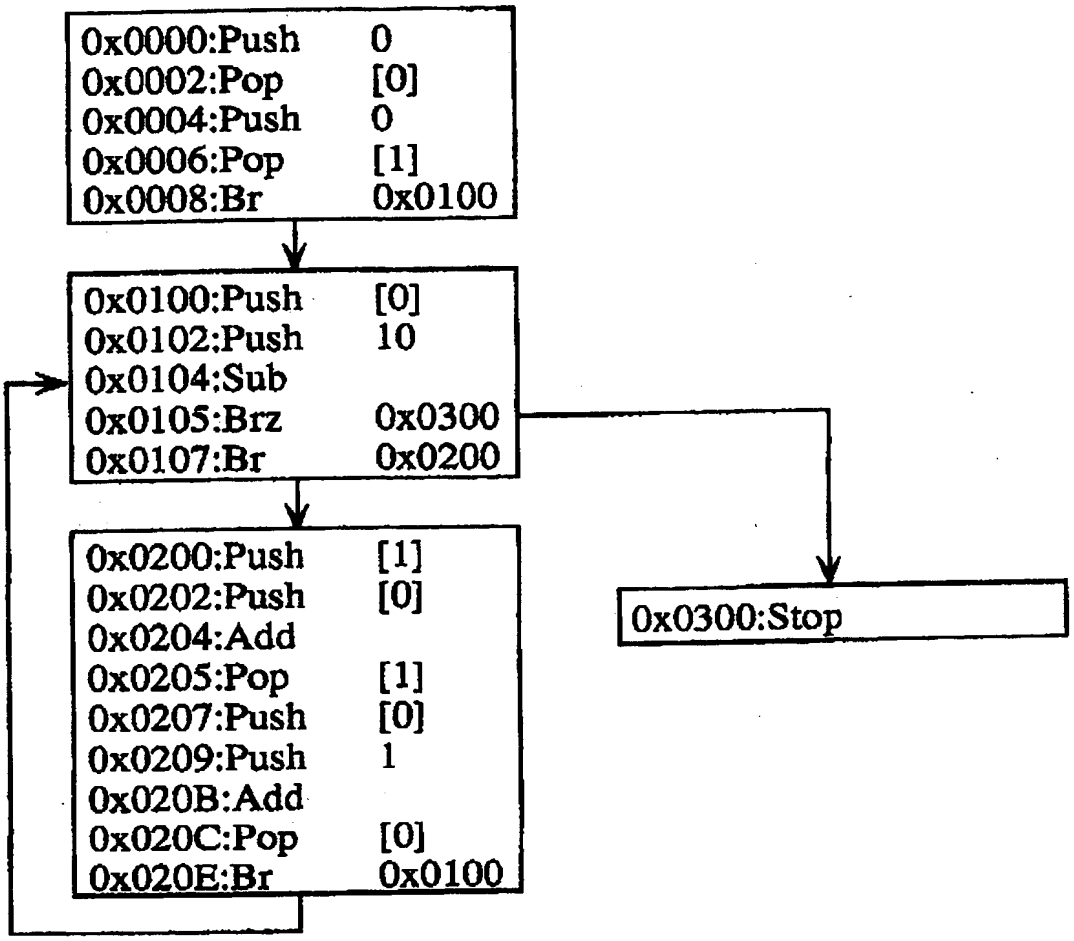
【図４１】


(ビット位置) 15 … 8 7 … 0
識別子セグメント
オフセット

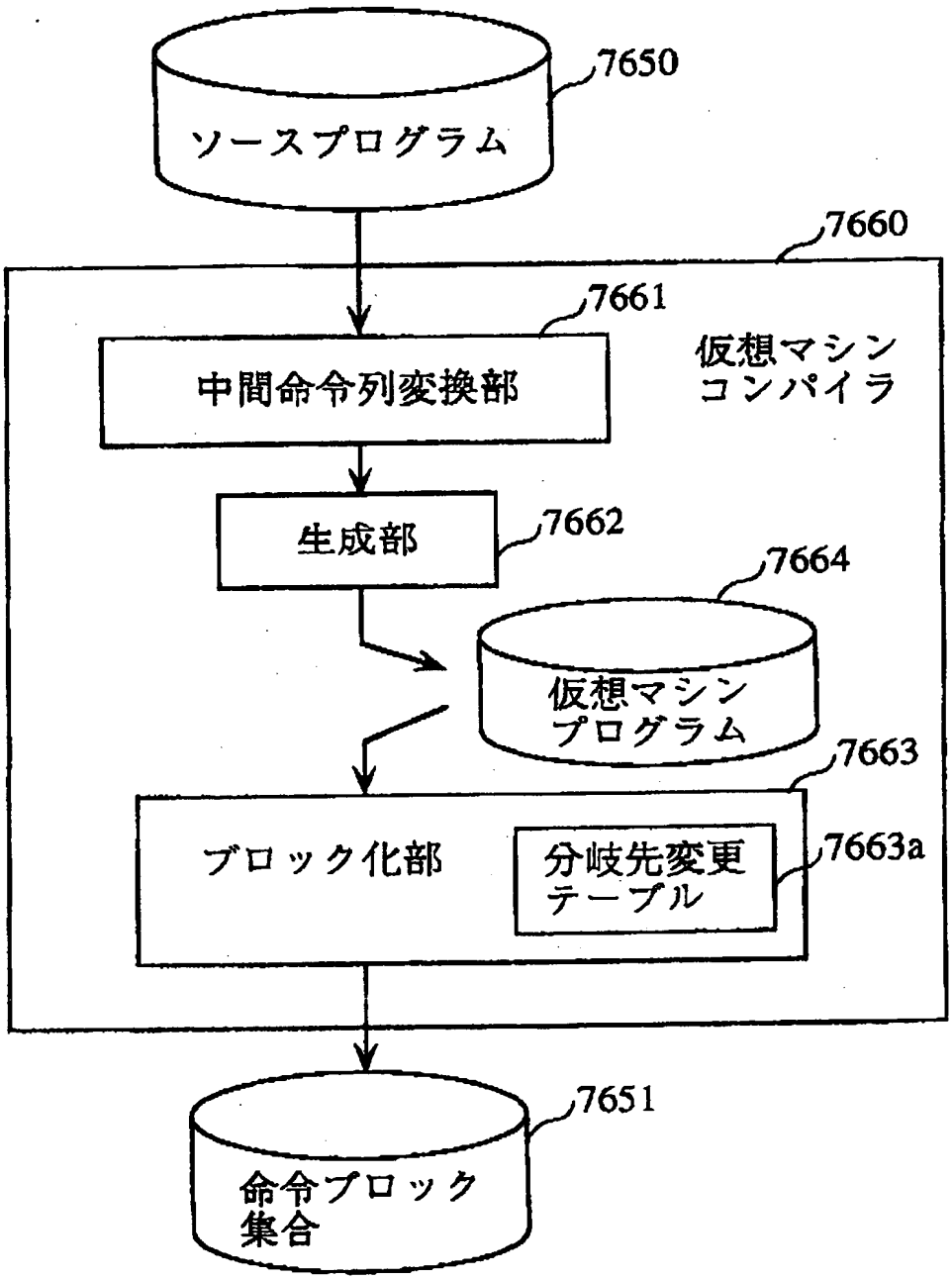
【図４２】


Start
8101
Jaddr←オペランド
8102
PCの識別子セグメント←Jaddr
PCのオフセット←0
8103
End
8104

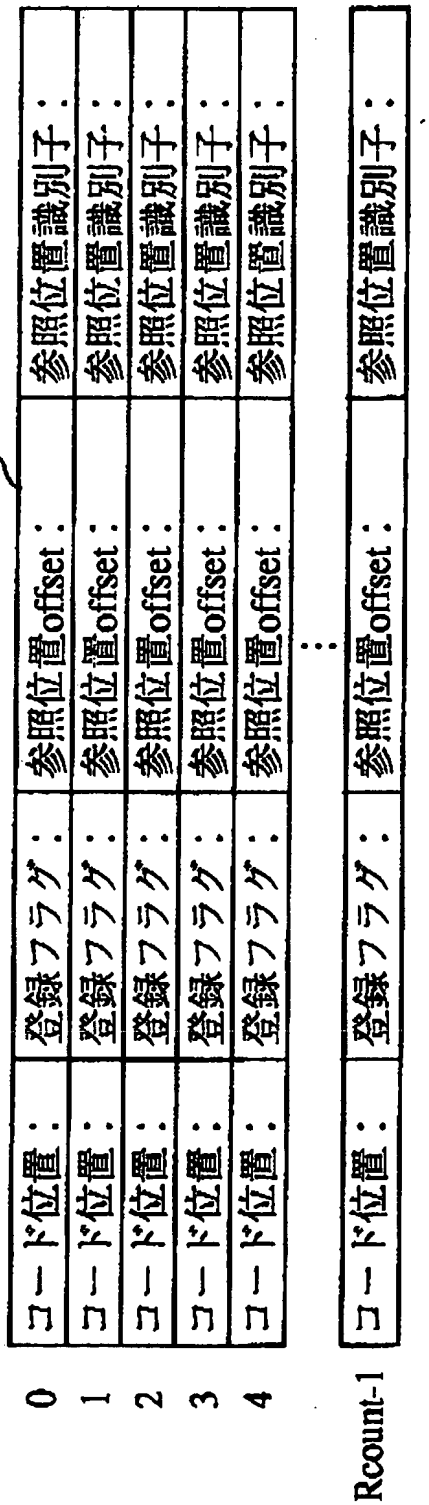
【図４３】


0x0000:Push 0
0x0002:Pop [0]
0x0004:Push 0
0x0006:Pop [1]
0x0008:Br 0x0100
0x0100:Push [0]
0x0102:Push 10
0x0104:Sub
0x0105:Brz 0x0300
0x0107:Br 0x0200
0x0200:Push [1]
0x0202:Push [0]
0x0204:Add
0x0205:Pop [1]
0x0207:Push [0]
0x0209:Push 1
0x020B:Add
0x020C:Pop [0]
0x020E:Br 0x0100
0x0300:Stop

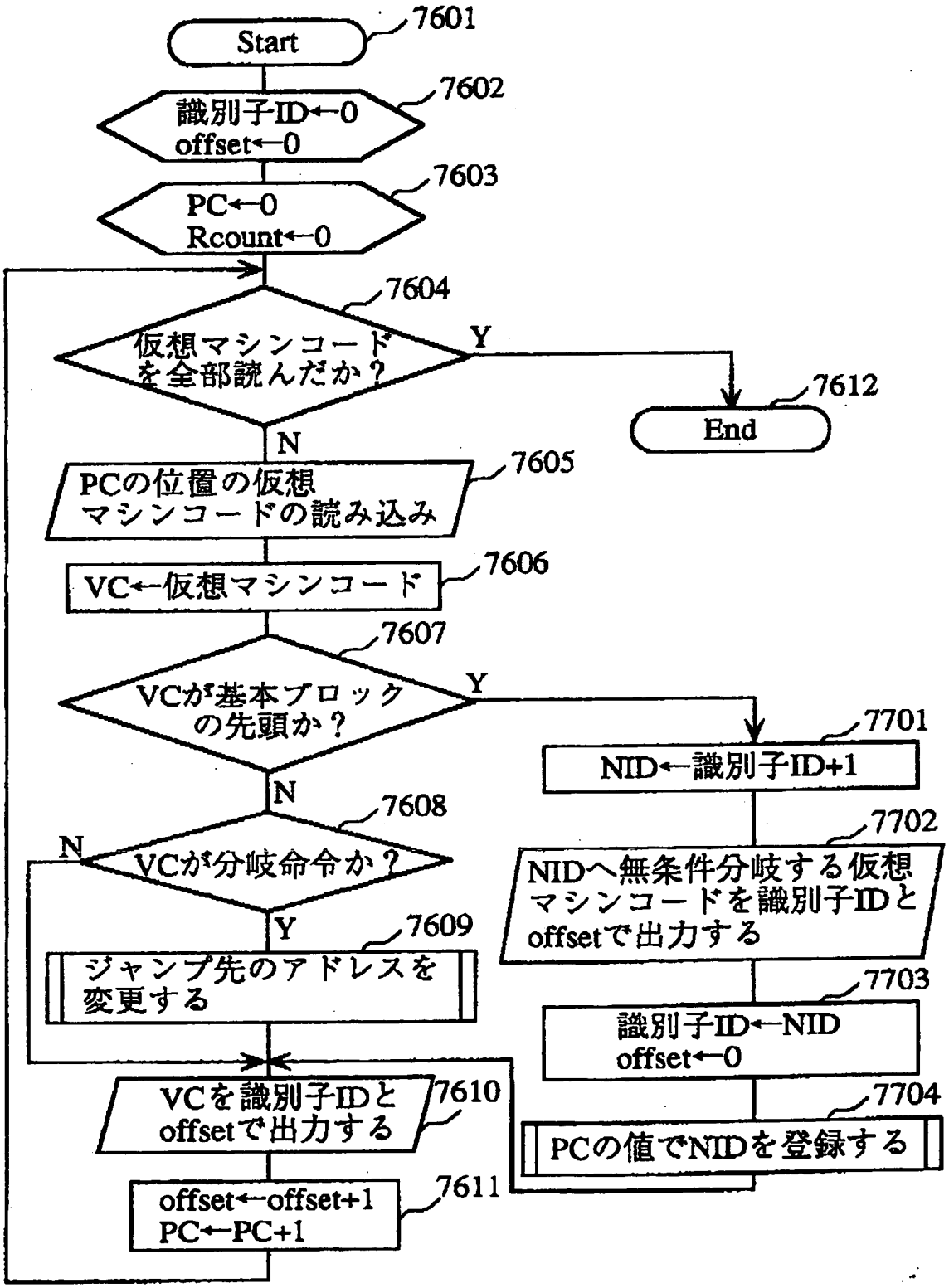
【図４４】


7650
ソースプログラム
7660
7661
中間命令列変換部
仮想マシン
コンパイラ
生成部
7662
7664
仮想マシン
プログラム
7663
ブロック化部
分岐先変更
テーブル
7663a
7651
命令ブロック
集合

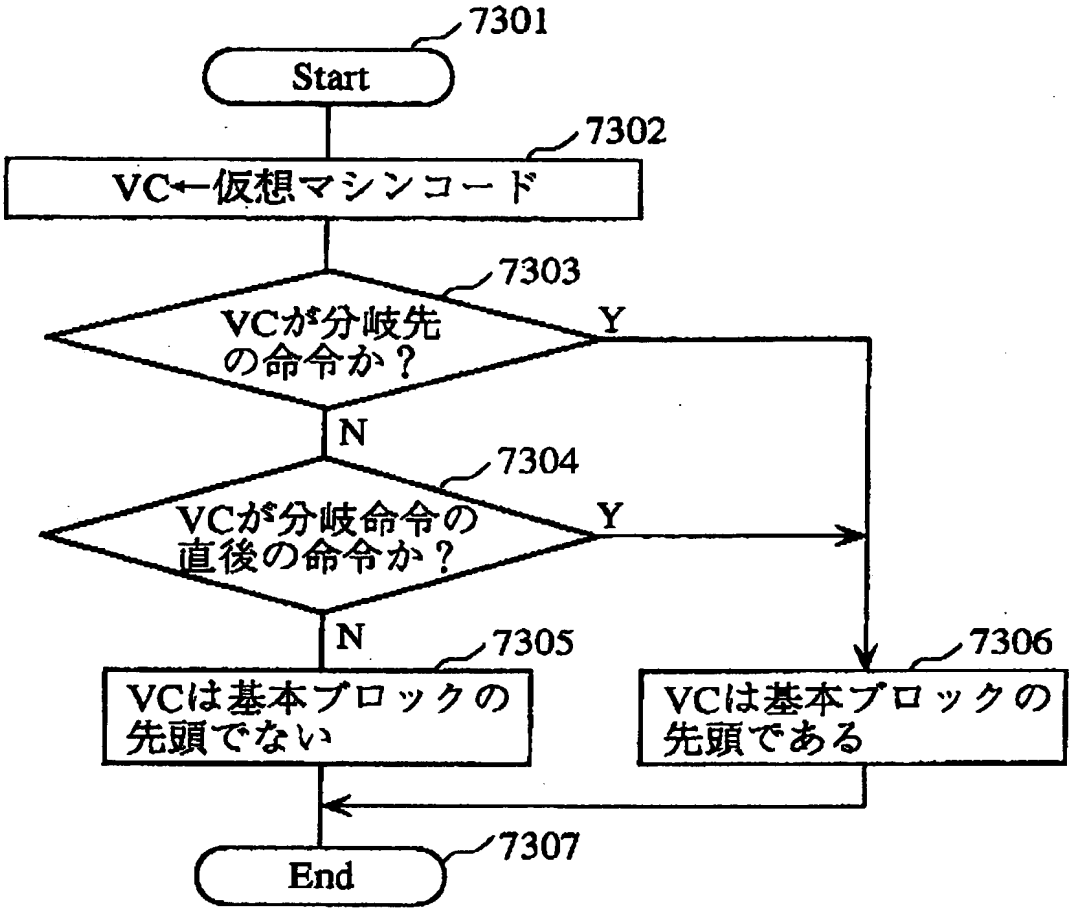
【図４５】



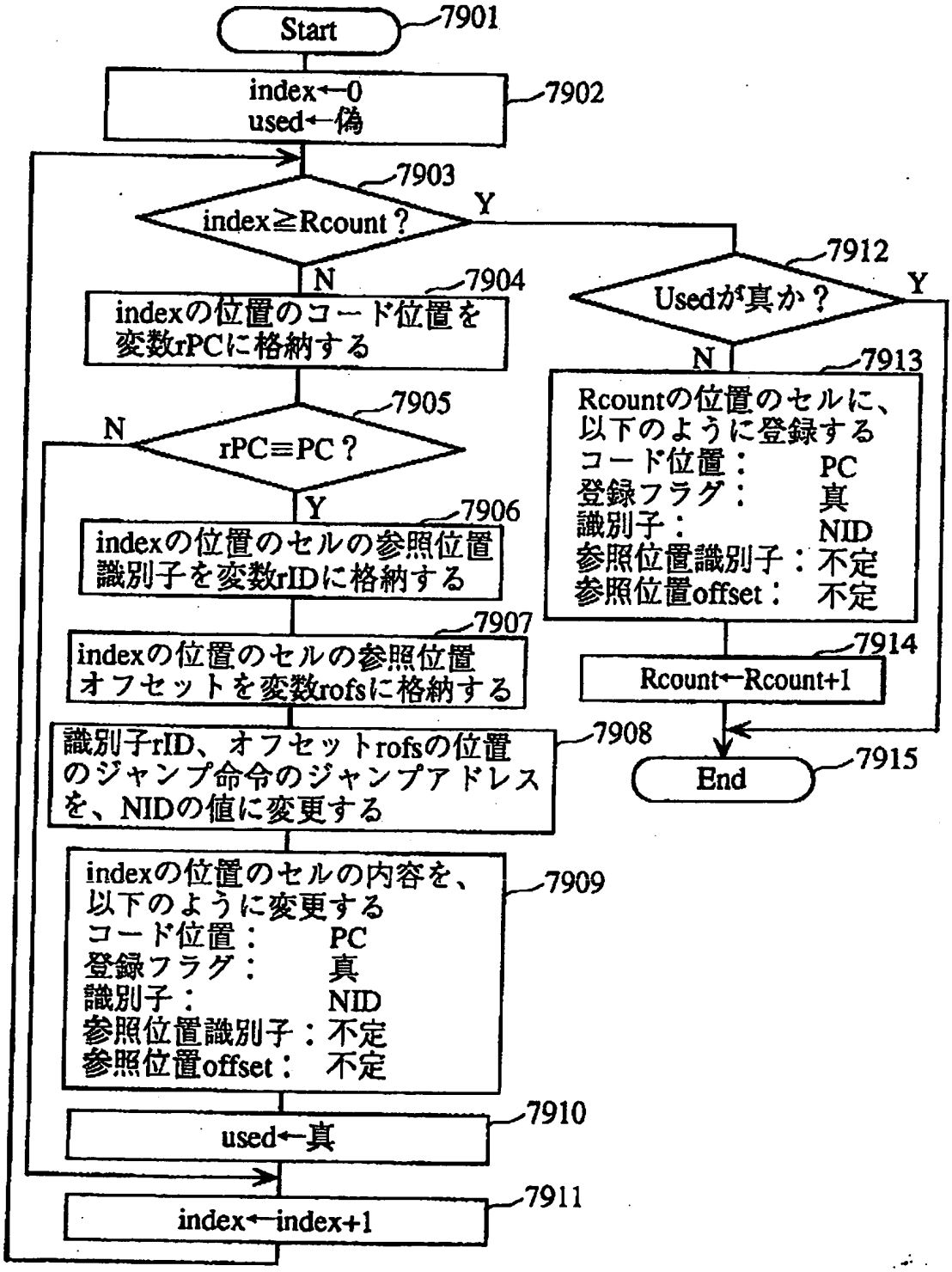
【図４６】


Start
7601
識別子ID←0
offset←0
7602
PC←0
Rcount←0
7603
仮想マシンコードを全部読んだか？
7604
Y
End
7612
N
PCの位置の仮想マシンコードの読み込み
7605
VC←仮想マシンコード
7606
VCが基本ブロックの先頭か？
7607
Y
NID←識別子ID+1
7701
N
VCが分岐命令か？
7608
N
Y
NIDへ無条件分岐する仮想マシンコードを識別子IDとoffsetで出力する
7702
ジャンプ先のアドレスを変更する
7609
識別子ID←NID
offset←0
7703
VCを識別子IDとoffsetで出力する
7610
PCの値でNIDを登録する
7704
offset←offset+1
PC←PC+1
7611

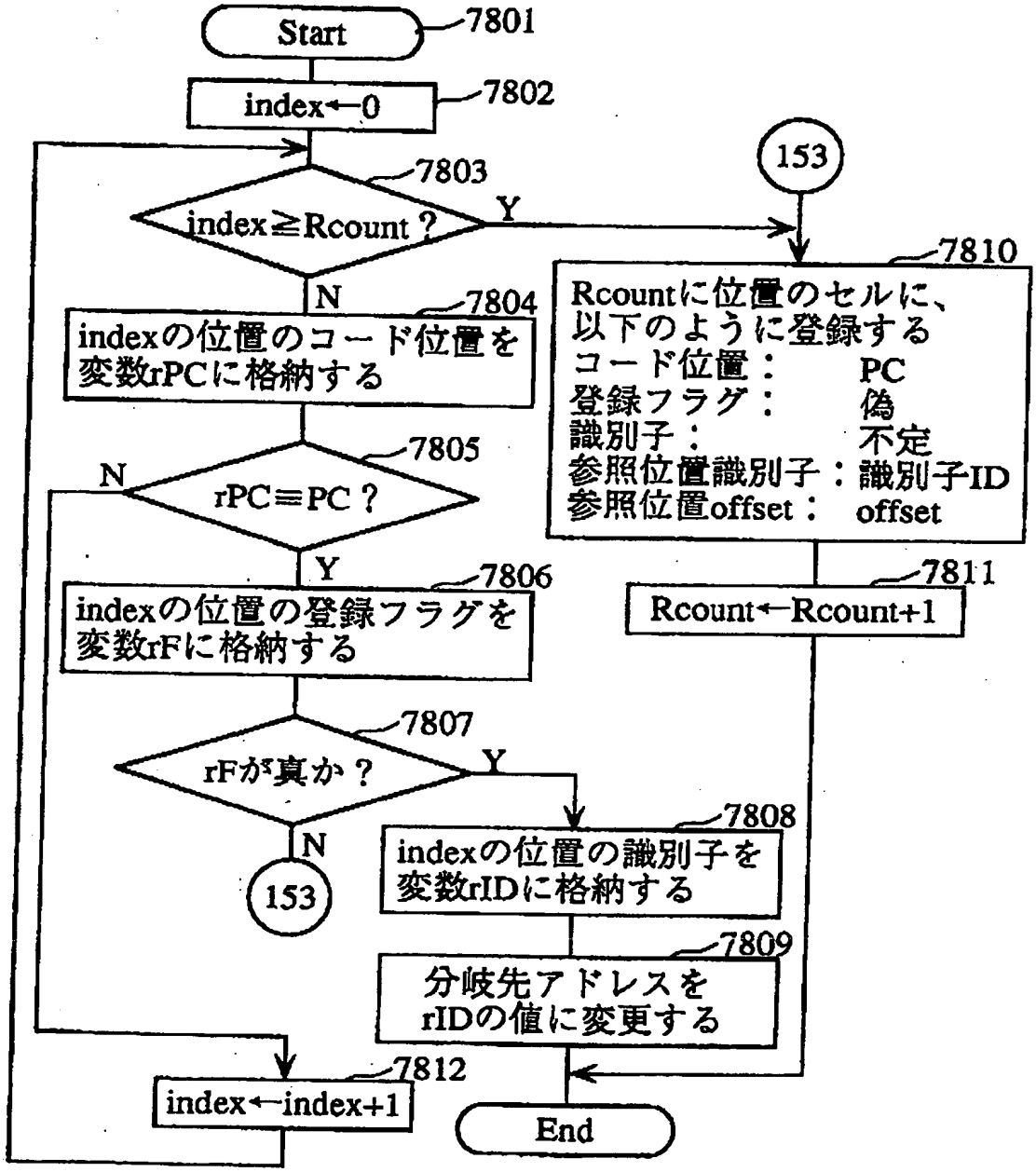
【図４７】


7301
Start
7302
VC←仮想マシンコード
7303
VCが分岐先の命令か？
Y
N
7304
VCが分岐命令の直後の命令か？
Y
N
7305
VCは基本ブロックの先頭でない
7306
VCは基本ブロックの先頭である
7307
End

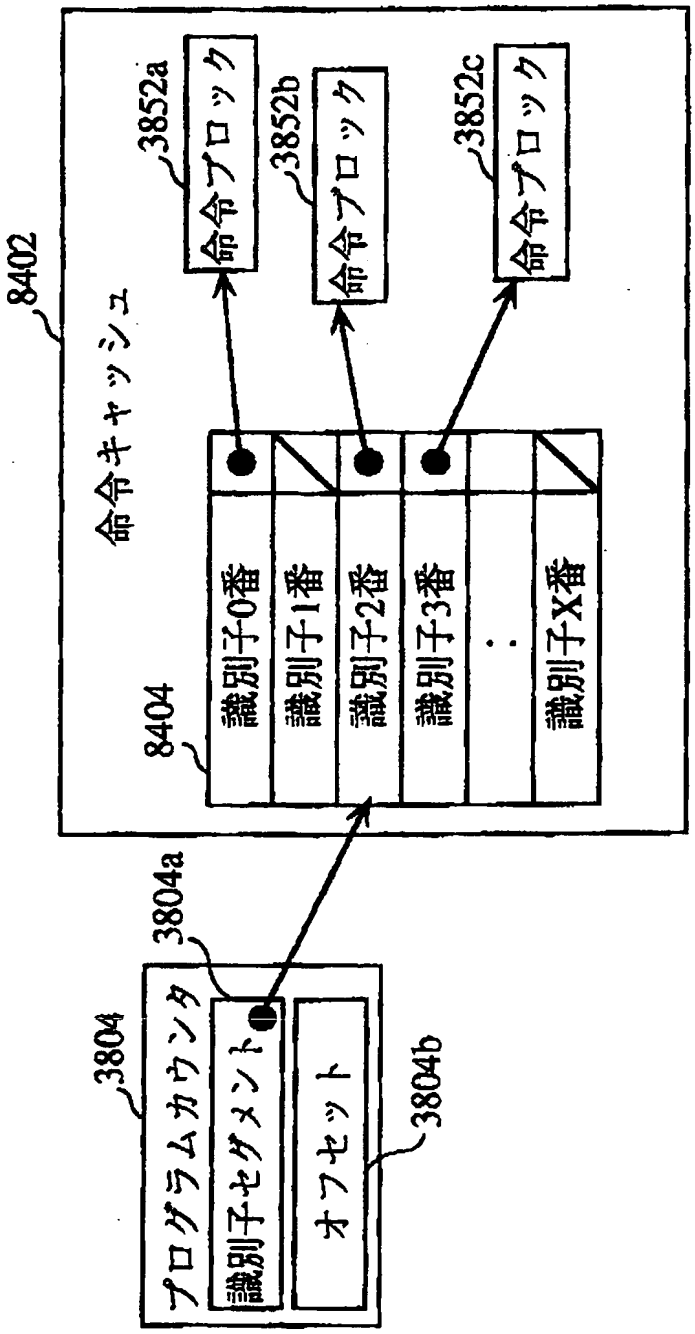
【図４８】


Start
7901
index←0
used←偽
7902
7903
index≧Rcount？
Y
N
7904
indexの位置のコード位置を
変数rPCに格納する
7905
N
rPC≡PC？
Y
7906
indexの位置のセルの参照位置
識別子を変数rIDに格納する
7907
indexの位置のセルの参照位置
オフセットを変数rofsに格納する
識別子rID、オフセットrofsの位置
のジャンプ命令のジャンプアドレス
を、NIDの値に変更する
7908
indexの位置のセルの内容を、
以下のように変更する
コード位置：　PC
登録フラグ：　真
識別子：　NID
参照位置識別子：不定
参照位置offset：　不定
7909
used←真
7910
index←index+1
7911
7912
Usedが真か？
Y
N
7913
Rcountの位置のセルに、
以下のように登録する
コード位置：　PC
登録フラグ：　真
識別子：　NID
参照位置識別子：不定
参照位置offset：　不定
7914
Rcount←Rcount+1
End
7915

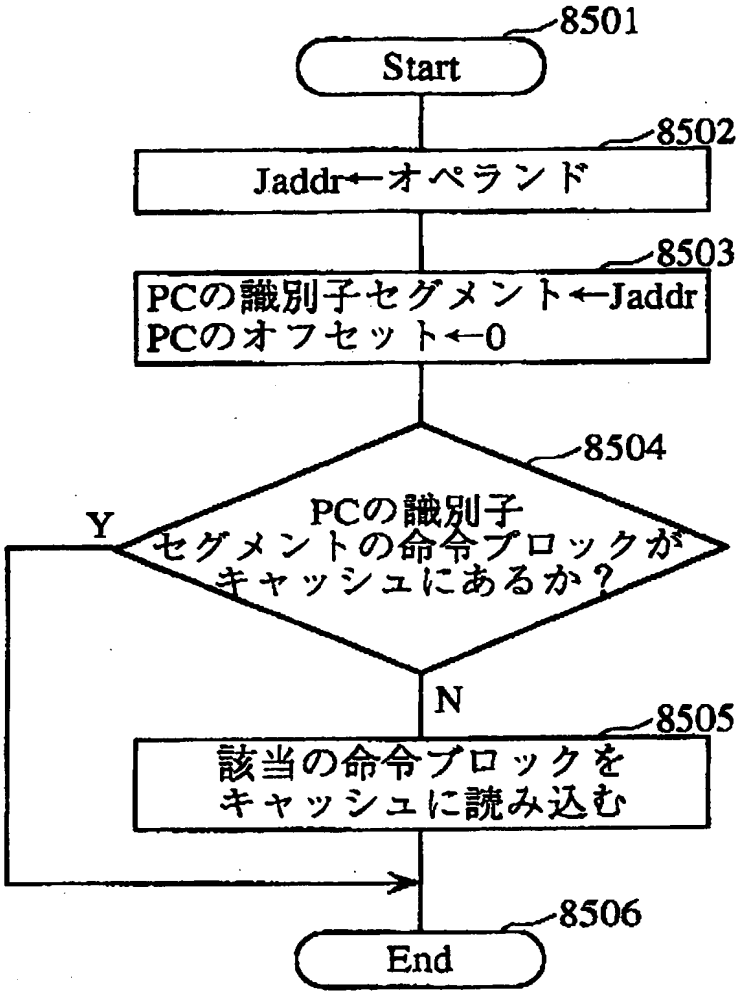
【図４９】


Start
7801
index←0
7802
7803
index≧Rcount？
Y
N
153
7810
Rcountに位置のセルに、
以下のように登録する
コード位置：　PC
登録フラグ：　偽
識別子：　不定
参照位置識別子：識別子ID
参照位置offset：　offset
7804
indexの位置のコード位置を
変数rPCに格納する
7805
N
rPC≡PC？
Y
7806
indexの位置の登録フラグを
変数rFに格納する
7811
Rcount←Rcount+1
7807
rFが真か？
Y
N
153
7808
indexの位置の識別子を
変数rIDに格納する
7809
分岐先アドレスを
rIDの値に変更する
7812
index←index+1
End

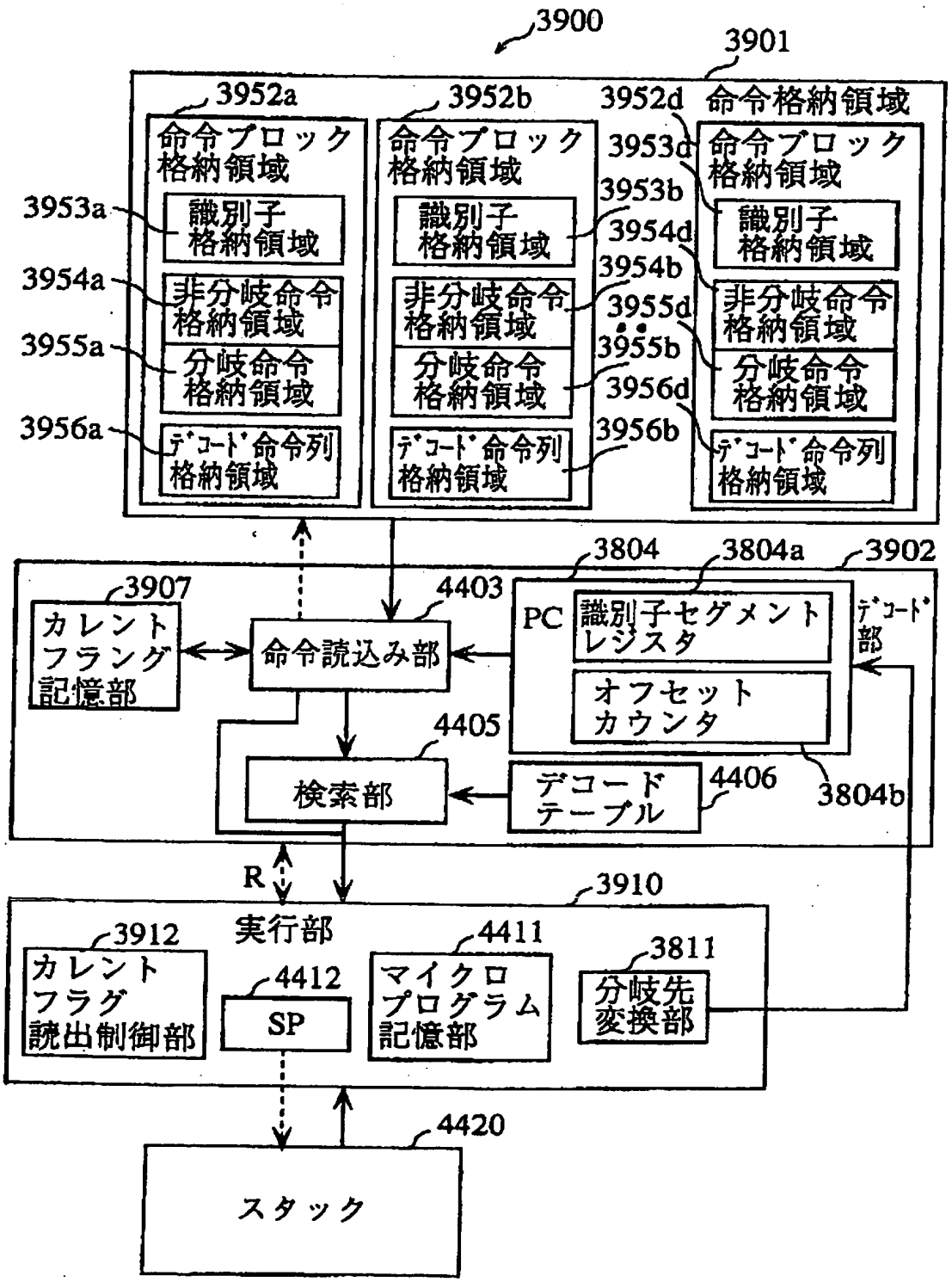
【図５０】


8402
命令キャッシュ
3852a
命令ブロック
3852b
命令ブロック
3852c
命令ブロック
8404
識別子0番
識別子1番
識別子2番
識別子3番
識別子X番
3804a
3804
プログラムカウンタ
識別子セグメント
オフセット
3804b

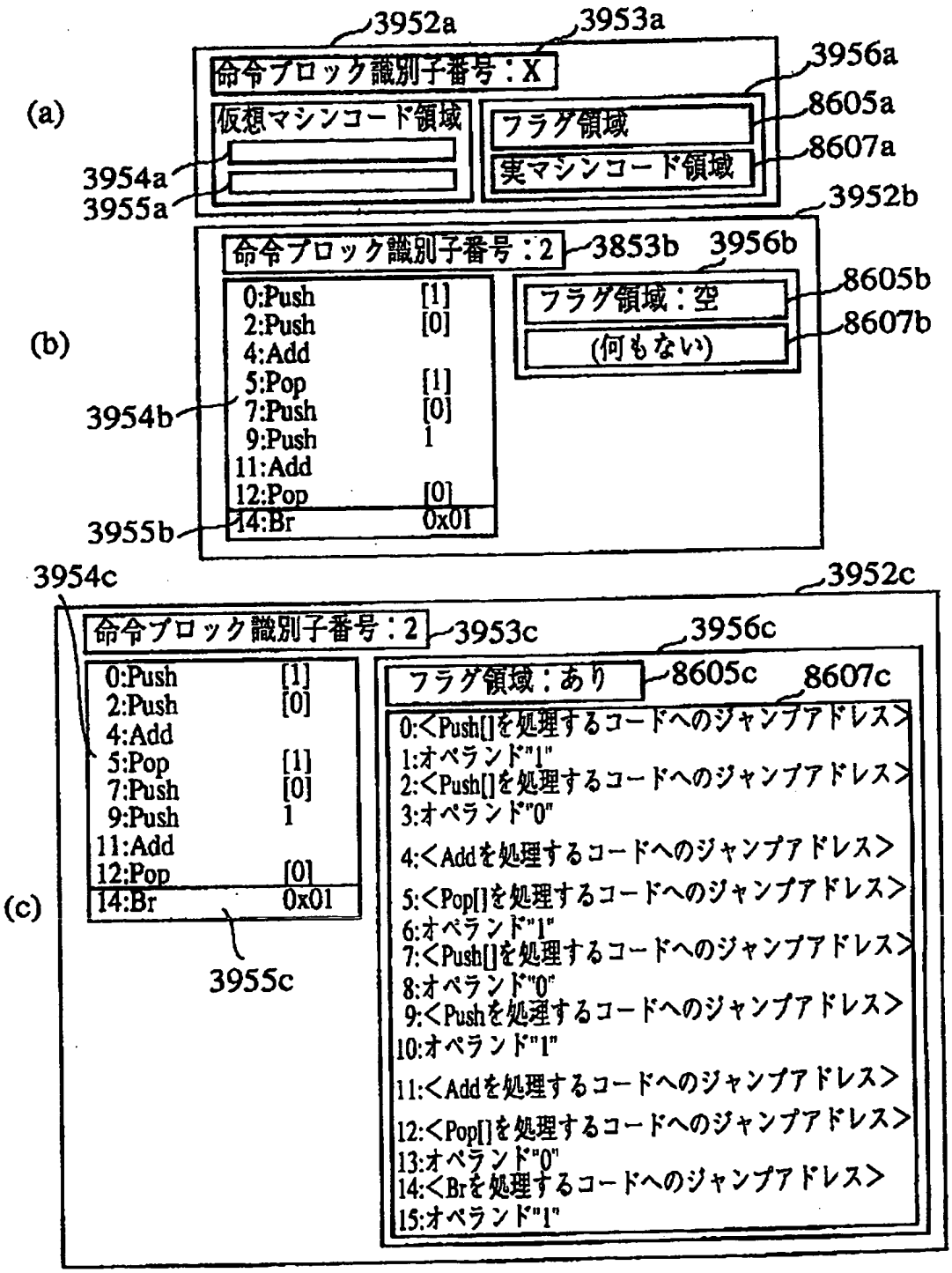
【図５１】


8501
Start
8502
Jaddr←オペランド
8503
PCの識別子セグメント←Jaddr
PCのオフセット←0
8504
PCの識別子
セグメントの命令ブロックが
キャッシュにあるか？
Y
N
8505
該当の命令ブロックを
キャッシュに読み込む
8506
End

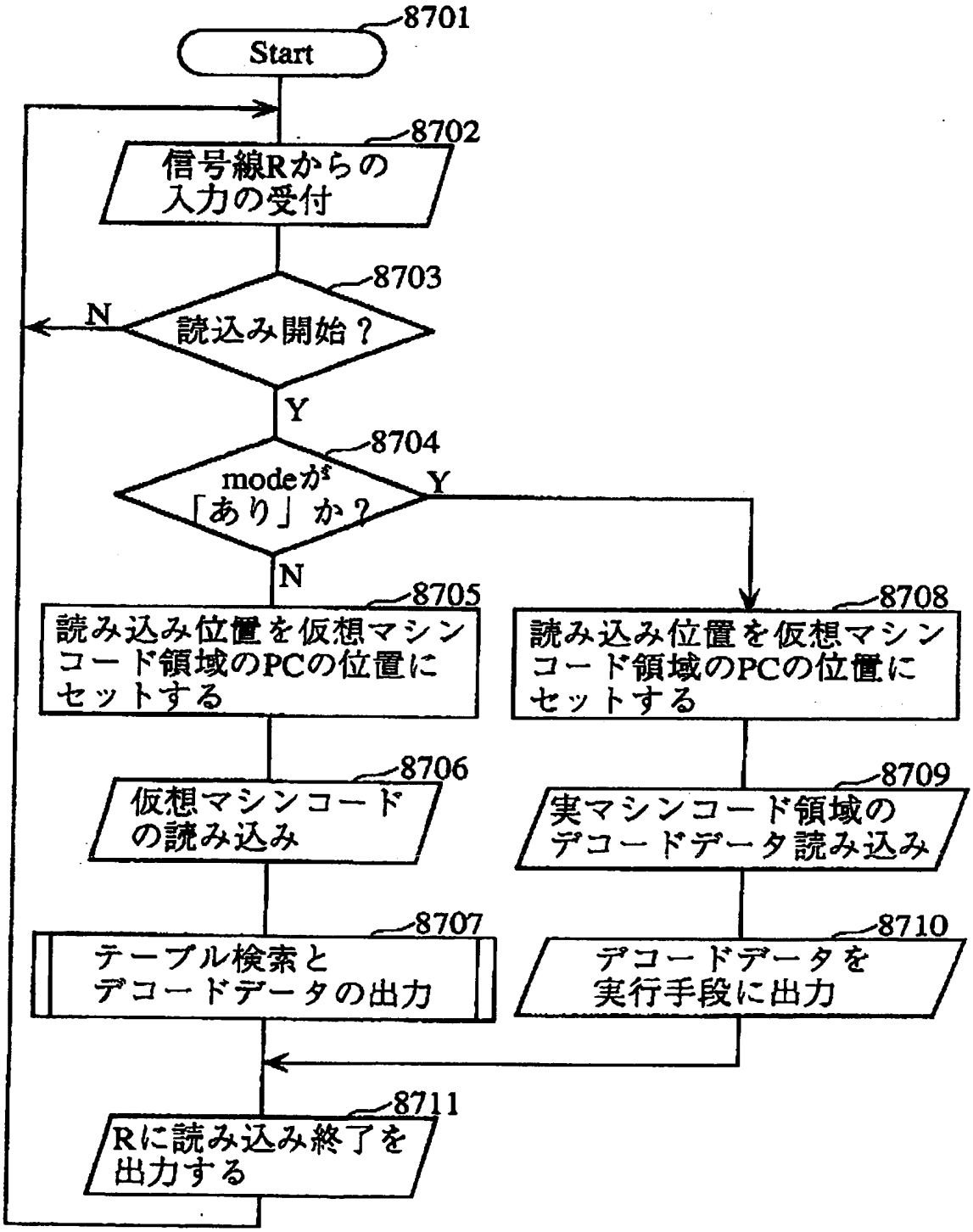
【図５２】


3900
3901
命令格納領域
3952a
3952b
3952d
命令ブロック格納領域
識別子格納領域
非分岐命令格納領域
分岐命令格納領域
ﾃﾞｺｰﾄﾞ命令列格納領域
3953a
3954a
3955a
3956a
3953b
3954b
3955b
3956b
3953d
3954d
3955d
3956d
3902
3804
3804a
3907
4403
PC
識別子セグメントレジスタ
オフセットカウンタ
ﾃﾞｺｰﾄﾞ部
カレントフラング記憶部
命令読込み部
4405
検索部
デコードテーブル
4406
3804b
R
3910
実行部
3912
カレントフラグ読出制御部
4412
SP
4411
マイクロプログラム記憶部
3811
分岐先変換部
4420
スタック

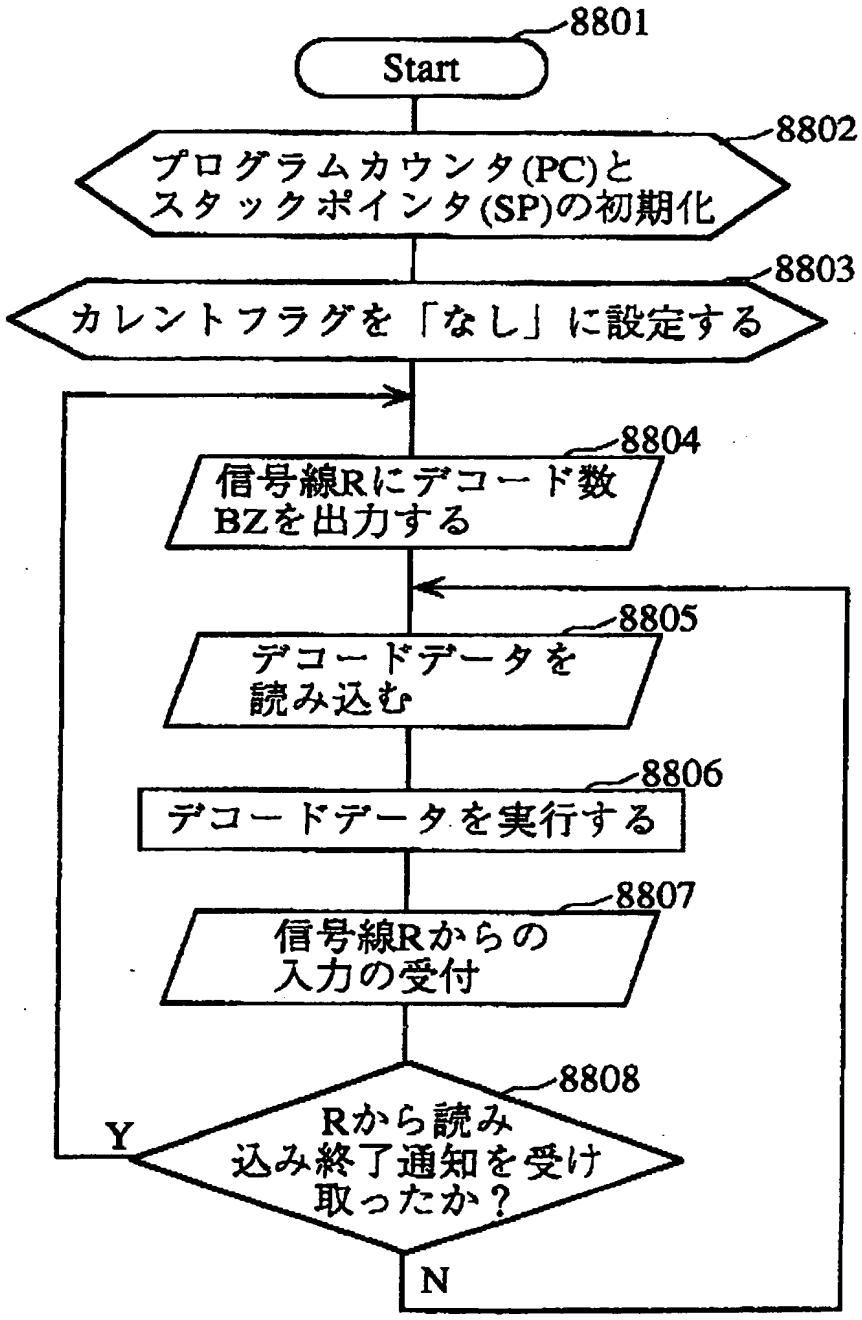
【図５３】


(a)
3952a
3953a
命令ブロック識別子番号：X
3956a
仮想マシンコード領域
フラグ領域
8605a
実マシンコード領域
8607a
3954a
3955a
(b)
3952b
命令ブロック識別子番号：2
3853b
3956b
0:Push [1]
2:Push [0]
4:Add
5:Pop [1]
7:Push [0]
9:Push 1
11:Add
12:Pop [0]
14:Br 0x01
フラグ領域：空
(何もない)
8605b
8607b
3954b
3955b
(c)
3954c
3952c
命令ブロック識別子番号：2
3953c
3956c
フラグ領域：あり
8605c
8607c
0:Push [1]
2:Push [0]
4:Add
5:Pop [1]
7:Push [0]
9:Push 1
11:Add
12:Pop [0]
14:Br 0x01
3955c
0:<Push[]を処理するコードへのジャンプアドレス>
1:オペランド"1"
2:<Push[]を処理するコードへのジャンプアドレス>
3:オペランド"0"
4:<Addを処理するコードへのジャンプアドレス>
5:<Pop[]を処理するコードへのジャンプアドレス>
6:オペランド"1"
7:<Push[]を処理するコードへのジャンプアドレス>
8:オペランド"0"
9:<Pushを処理するコードへのジャンプアドレス>
10:オペランド"1"
11:<Addを処理するコードへのジャンプアドレス>
12:<Pop[]を処理するコードへのジャンプアドレス>
13:オペランド"0"
14:<Brを処理するコードへのジャンプアドレス>
15:オペランド"1"

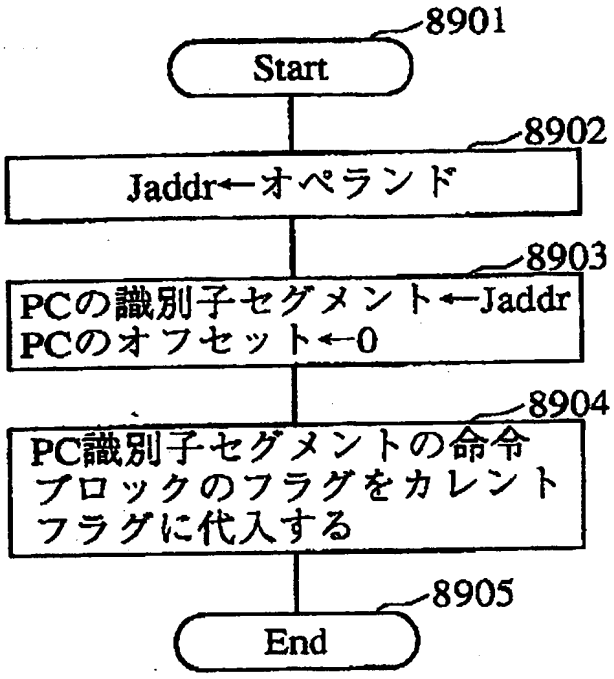
【図５４】


8701
Start
8702
信号線Rからの
入力の受付
8703
読込み開始？
N
Y
8704
modeが
「あり」か？
Y
N
8705
読み込み位置を仮想マシン
コード領域のPCの位置に
セットする
8706
仮想マシンコード
の読み込み
8707
テーブル検索と
デコードデータの出力
8708
読み込み位置を仮想マシン
コード領域のPCの位置に
セットする
8709
実マシンコード領域の
デコードデータ読み込み
8710
デコードデータを
実行手段に出力
8711
Rに読み込み終了を
出力する

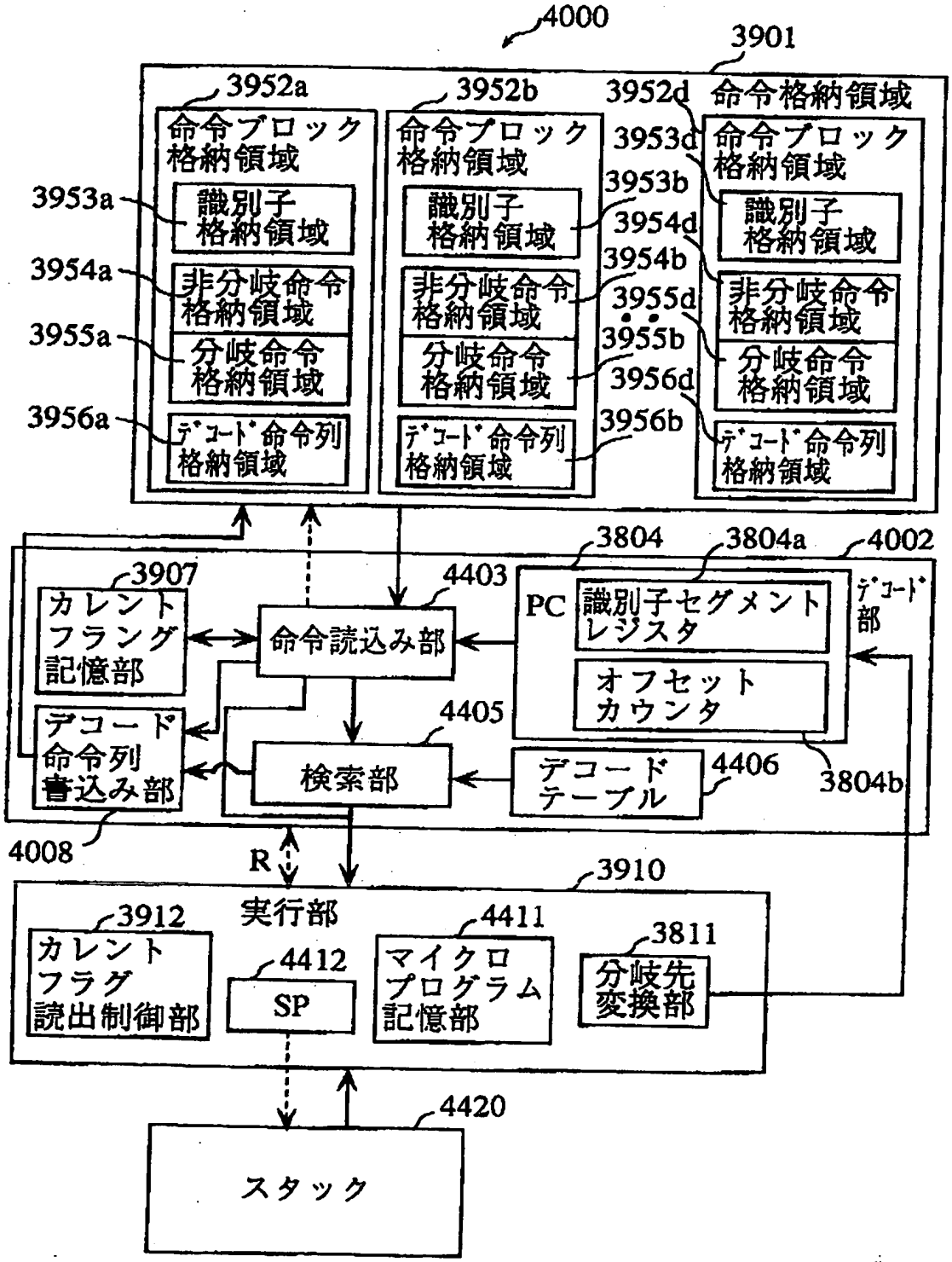
【図５５】


8801
Start
8802
プログラムカウンタ(PC)と
スタックポインタ(SP)の初期化
8803
カレントフラグを「なし」に設定する
8804
信号線Rにデコード数
BZを出力する
8805
デコードデータを
読み込む
8806
デコードデータを実行する
8807
信号線Rからの
入力の受付
8808
Rから読み
込み終了通知を受け
取ったか？
Y
N

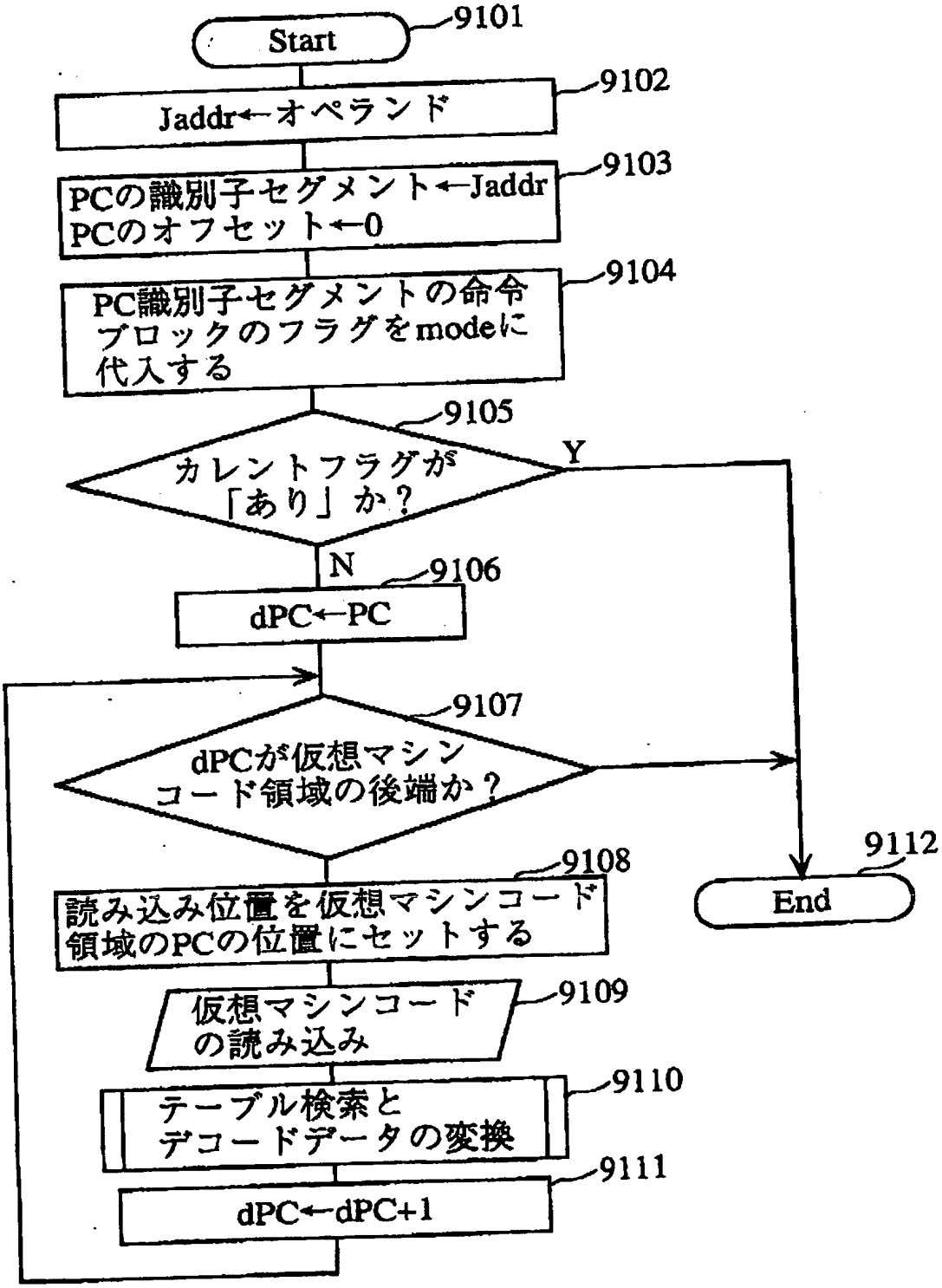
【図５６】


8901
Start
8902
Jaddr←オペランド
8903
PCの識別子セグメント←Jaddr
PCのオフセット←0
8904
PC識別子セグメントの命令
ブロックのフラグをカレント
フラグに代入する
8905
End

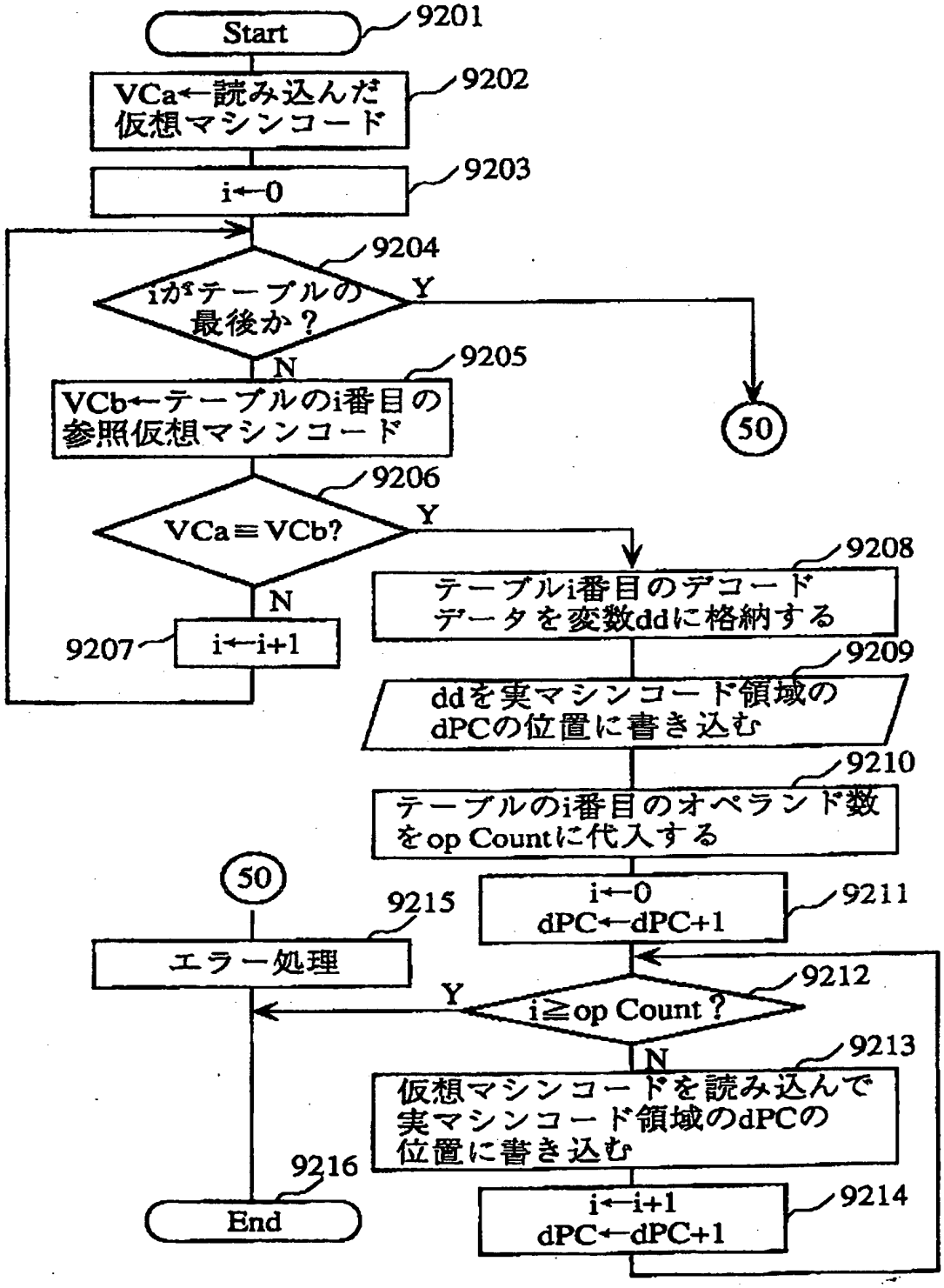
【図５７】


4000
3901
3952a
命令ブロック格納領域
3952b
命令ブロック格納領域
3952d
命令格納領域
命令ブロック格納領域
3953a
識別子格納領域
3953b
識別子格納領域
3953d
識別子格納領域
3954a
非分岐命令格納領域
3954b
非分岐命令格納領域
3954d
非分岐命令格納領域
3955a
分岐命令格納領域
3955b
分岐命令格納領域
3955d
分岐命令格納領域
3956a
ﾃﾞｺｰﾄﾞ命令列格納領域
3956b
ﾃﾞｺｰﾄﾞ命令列格納領域
3956d
ﾃﾞｺｰﾄﾞ命令列格納領域
3804
3804a
4002
3907
カレントフラング記憶部
4403
命令読込み部
PC
識別子セグメントレジスタ
ﾃﾞｺｰﾄﾞ部
オフセットカウンタ
4405
デコード命令列書込み部
検索部
デコードテーブル
4406
3804b
4008
R
3910
実行部
3912
カレントフラグ読出制御部
4411
マイクロプログラム記憶部
3811
分岐先変換部
4412
SP
4420
スタック

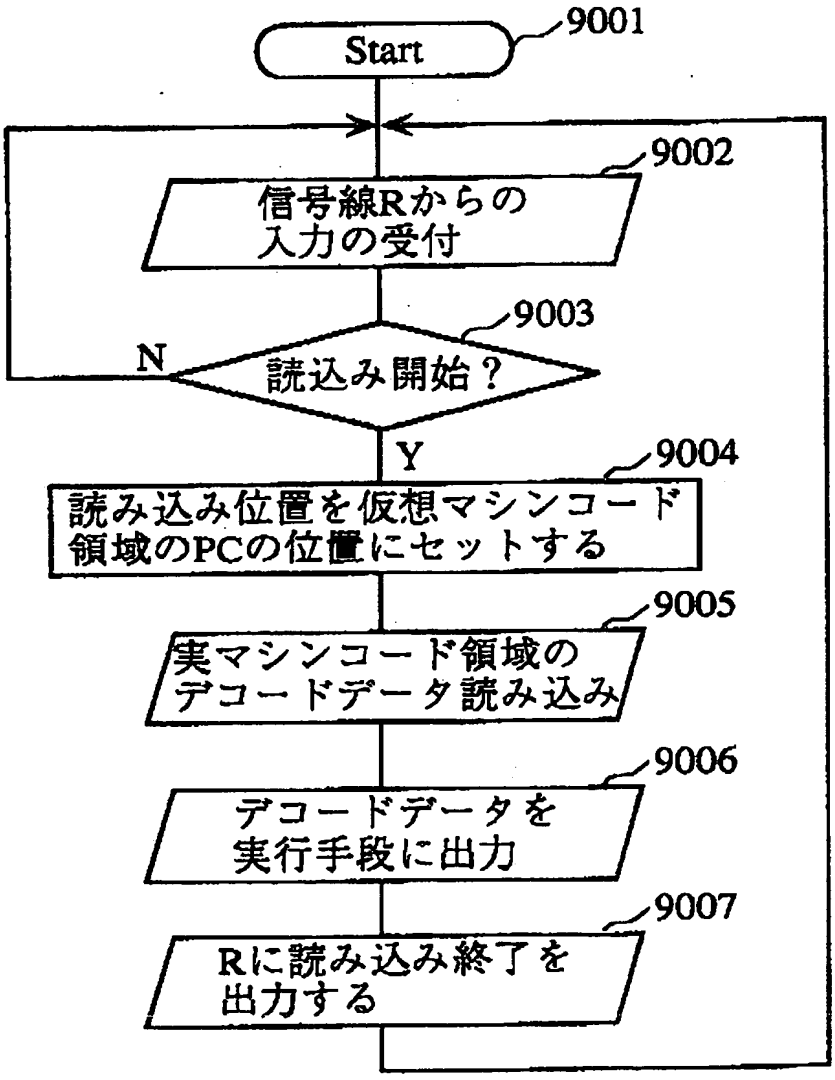
【図５８】


Start
9101
Jaddr←オペランド
9102
PCの識別子セグメント←Jaddr
PCのオフセット←0
9103
PC識別子セグメントの命令
ブロックのフラグをmodeに
代入する
9104
カレントフラグが
「あり」か？
9105
Y
N
dPC←PC
9106
dPCが仮想マシン
コード領域の後端か？
9107
読み込み位置を仮想マシンコード
領域のPCの位置にセットする
9108
仮想マシンコード
の読み込み
9109
テーブル検索と
デコードデータの変換
9110
dPC←dPC+1
9111
End
9112

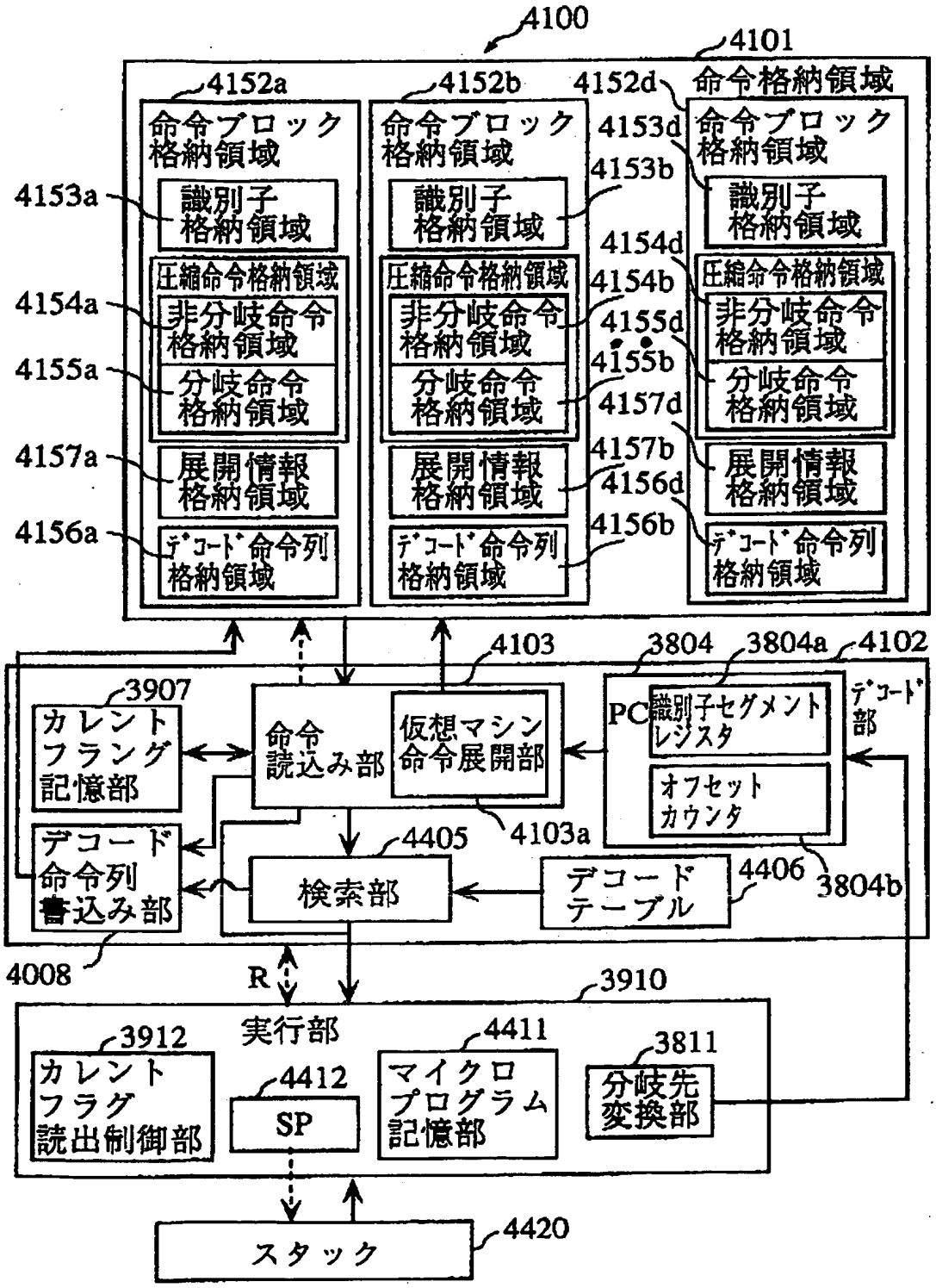
【図５９】


Start
9201
VCa←読み込んだ仮想マシンコード
9202
i←0
9203
9204
iがテーブルの最後か？
Y
N
9205
VCb←テーブルのi番目の参照仮想マシンコード
50
9206
VCa≡VCb?
Y
N
9207
i←i+1
9208
テーブルi番目のデコードデータを変数ddに格納する
9209
ddを実マシンコード領域のdPCの位置に書き込む
9210
テーブルのi番目のオペランド数をop Countに代入する
50
9215
エラー処理
i←0
dPC←dPC+1
9211
9212
i≧op Count？
Y
N
9213
仮想マシンコードを読み込んで実マシンコード領域のdPCの位置に書き込む
9216
End
i←i+1
dPC←dPC+1
9214

【図６０】


Start
9001
9002
信号線Rからの
入力の受付
9003
N
読込み開始？
Y
9004
読み込み位置を仮想マシンコード
領域のPCの位置にセットする
9005
実マシンコード領域の
デコードデータ読み込み
9006
デコードデータを
実行手段に出力
9007
Rに読み込み終了を
出力する

【図６１】

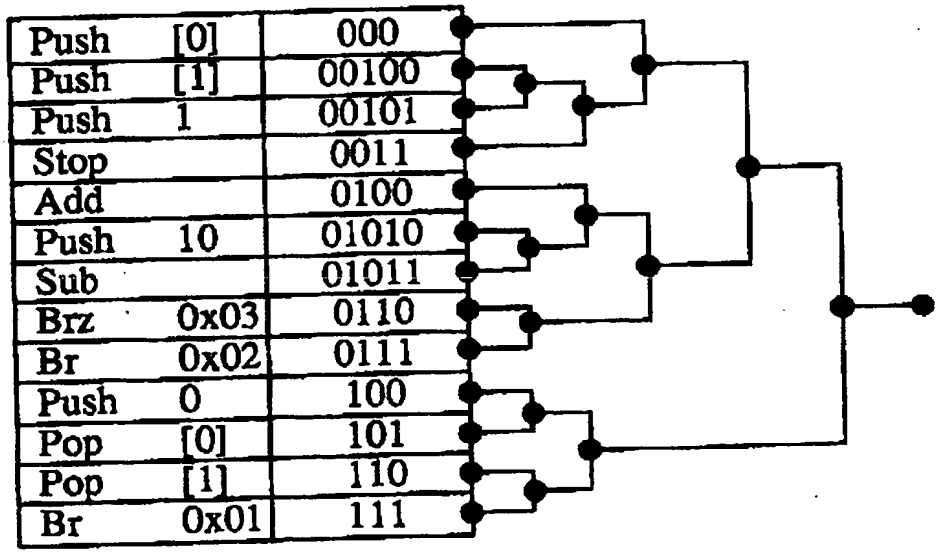

4100
4101
命令格納領域
4152a
4152b
4152d
命令ブロック格納領域
識別子格納領域
圧縮命令格納領域
非分岐命令格納領域
分岐命令格納領域
展開情報格納領域
ﾃﾞｺｰﾄﾞ命令列格納領域
4153a
4153b
4153d
4154a
4154b
4154d
4155a
4155b
4155d
4157a
4157b
4157d
4156a
4156b
4156d
4103
3804
3804a
4102
3907
カレントフラング記憶部
命令読込み部
仮想マシン命令展開部
PC
識別子セグメントレジスタ
オフセットカウンタ
ﾃﾞｺｰﾄﾞ部
4103a
4405
4406
3804b
デコード命令列書込み部
検索部
デコードテーブル
4008
R
3910
実行部
3912
カレントフラグ読出制御部
4412
SP
4411
マイクロプログラム記憶部
3811
分岐先変換部
スタック
4420

【図６２】

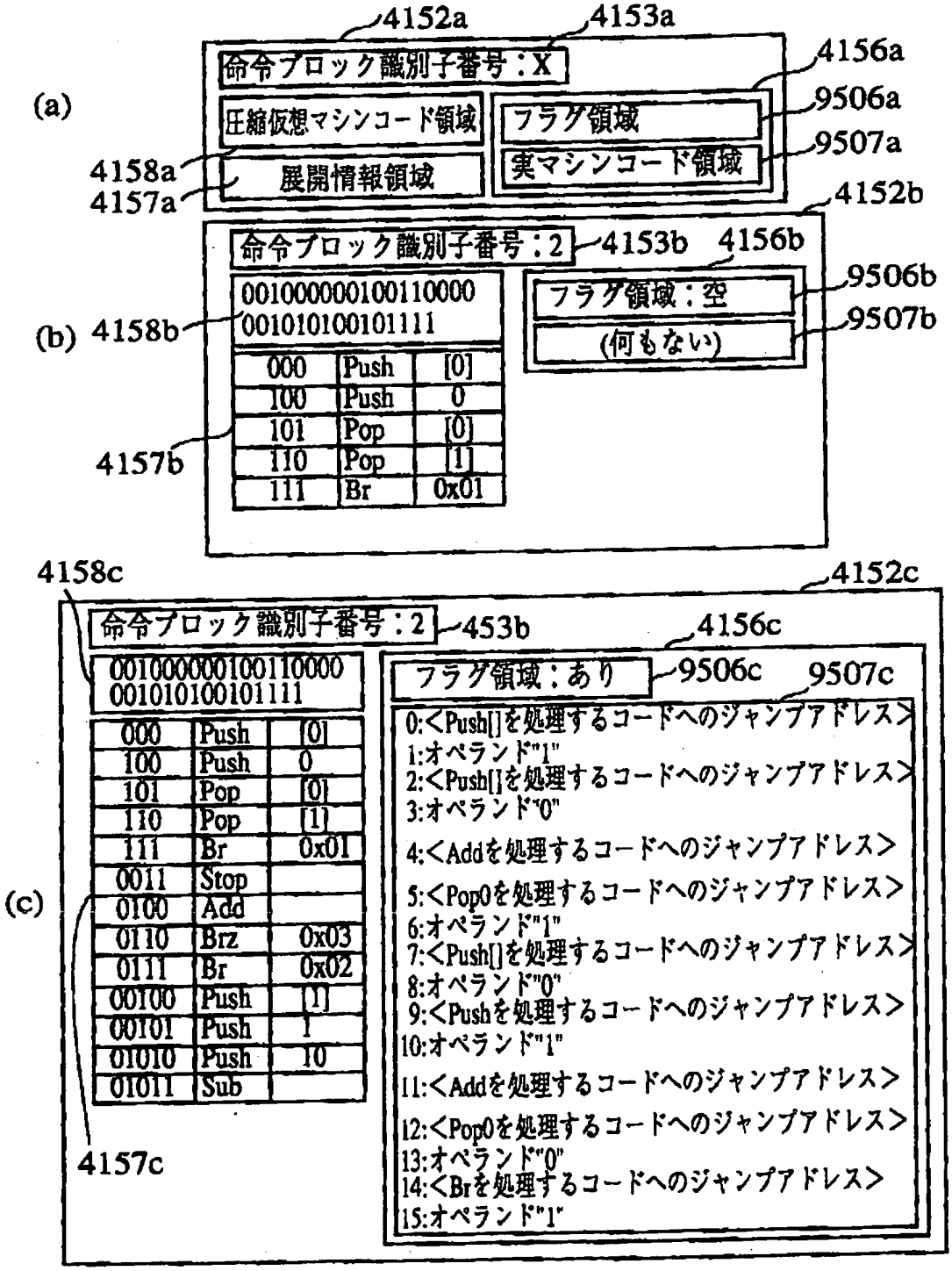
(a)

ビット列	意味
000	Push [0]
100	Push 0
101	Pop [0]
110	Pop [1]
111	Br 0x01
0011	Stop
0100	Add
0110	Brz 0x03
0111	Br 0x02
00100	Push [1]
00101	Push 1
01010	Push 10
01011	Sub

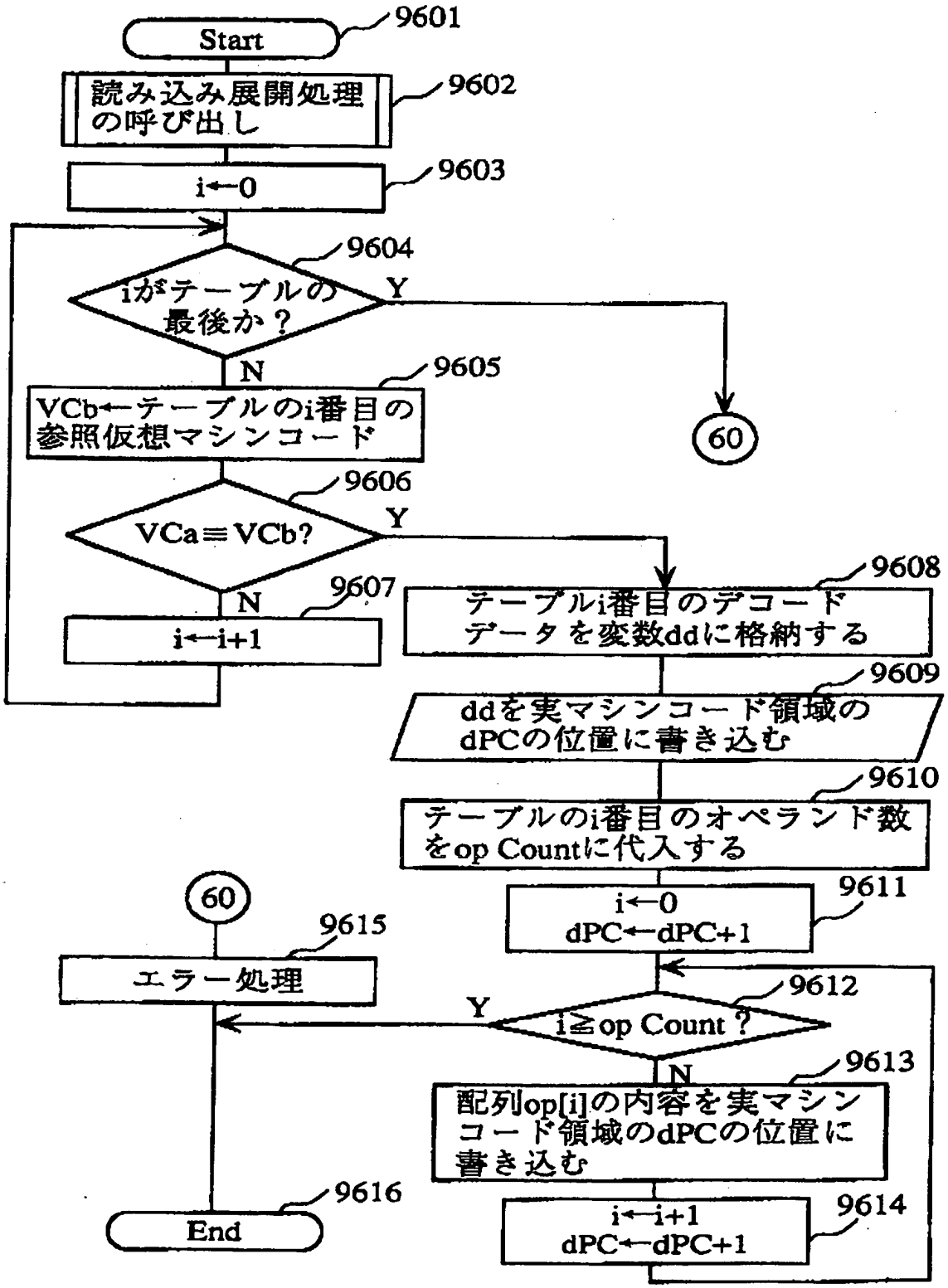
(b)



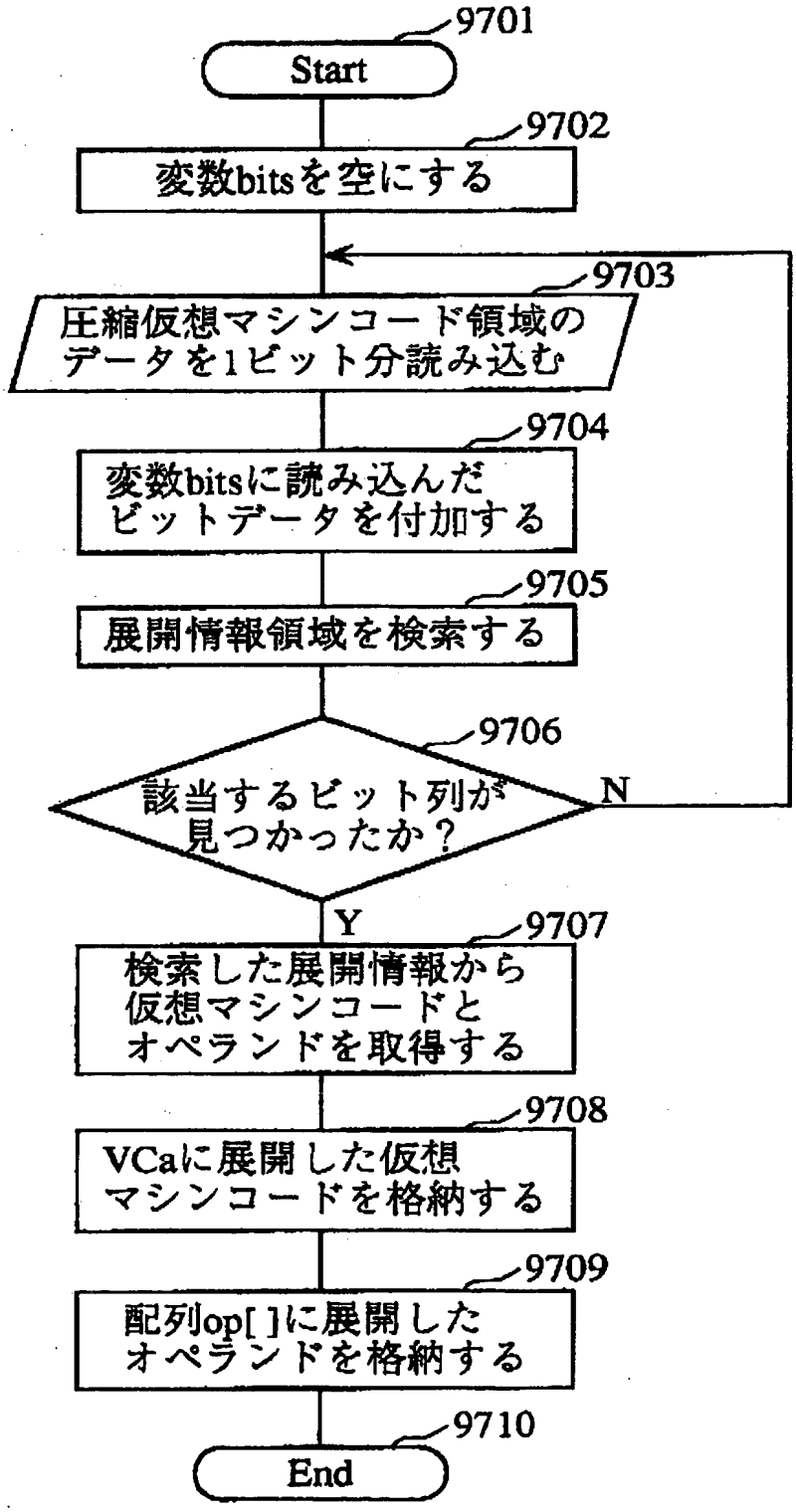
【図６３】


(a)
4152a
4153a
命令ブロック識別子番号：X
4156a
圧縮仮想マシンコード領域
フラグ領域
9506a
9507a
実マシンコード領域
4158a
展開情報領域
4157a
4152b
命令ブロック識別子番号：2
4153b
4156b
(b)
4158b
0010000001001100000010101001011111
フラグ領域：空
9506b
9507b
(何もない)
000 Push [0]
100 Push 0
101 Pop [0]
110 Pop [1]
111 Br 0x01
4157b
4158c
4152c
命令ブロック識別子番号：2
453b
4156c
0010000001001100000010101001011111
フラグ領域：あり
9506c
9507c
000 Push [0]
100 Push 0
101 Pop [0]
110 Pop [1]
111 Br 0x01
0011 Stop
(c)
0100 Add
0110 Brz 0x03
0111 Br 0x02
00100 Push [1]
00101 Push 1
01010 Push 10
01011 Sub
4157c
0:<Push[]を処理するコードへのジャンプアドレス>
1:オペランド"1"
2:<Push[]を処理するコードへのジャンプアドレス>
3:オペランド"0"
4:<Addを処理するコードへのジャンプアドレス>
5:<Pop0を処理するコードへのジャンプアドレス>
6:オペランド"1"
7:<Push[]を処理するコードへのジャンプアドレス>
8:オペランド"0"
9:<Pushを処理するコードへのジャンプアドレス>
10:オペランド"1"
11:<Addを処理するコードへのジャンプアドレス>
12:<Pop0を処理するコードへのジャンプアドレス>
13:オペランド"0"
14:<Brを処理するコードへのジャンプアドレス>
15:オペランド"1"

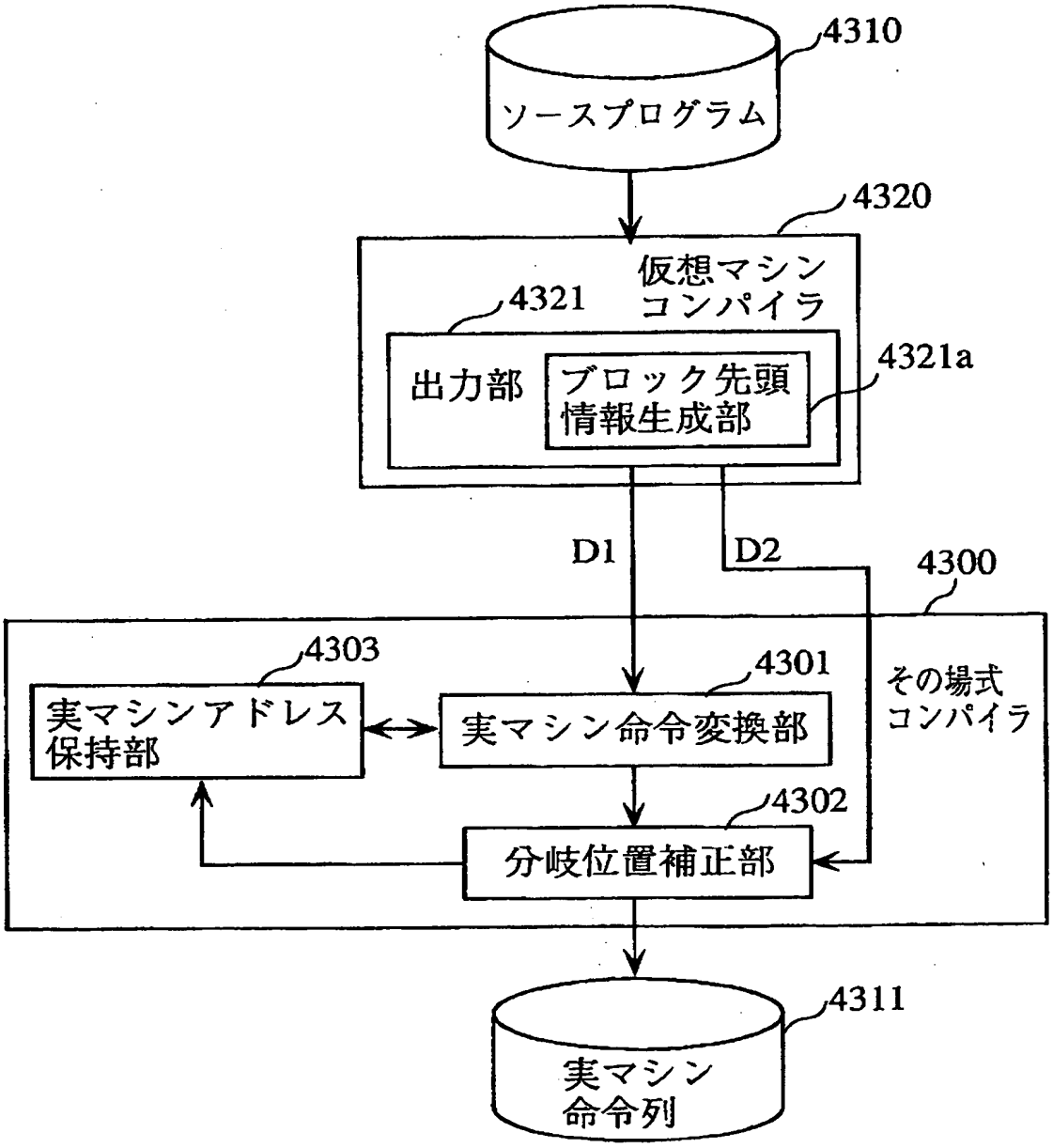
【図６４】


Start
9601
読み込み展開処理の呼び出し
9602
i←0
9603
9604
iがテーブルの最後か？
Y
N
9605
VCb←テーブルのi番目の参照仮想マシンコード
60
9606
VCa≡VCb?
Y
N
9607
i←i+1
9608
テーブルi番目のデコードデータを変数ddに格納する
9609
ddを実マシンコード領域のdPCの位置に書き込む
9610
テーブルのi番目のオペランド数をop Countに代入する
9611
i←0
dPC←dPC+1
60
9615
エラー処理
9612
Y
i≧op Count？
N
9613
配列op[i]の内容を実マシンコード領域のdPCの位置に書き込む
9614
i←i+1
dPC←dPC+1
9616
End

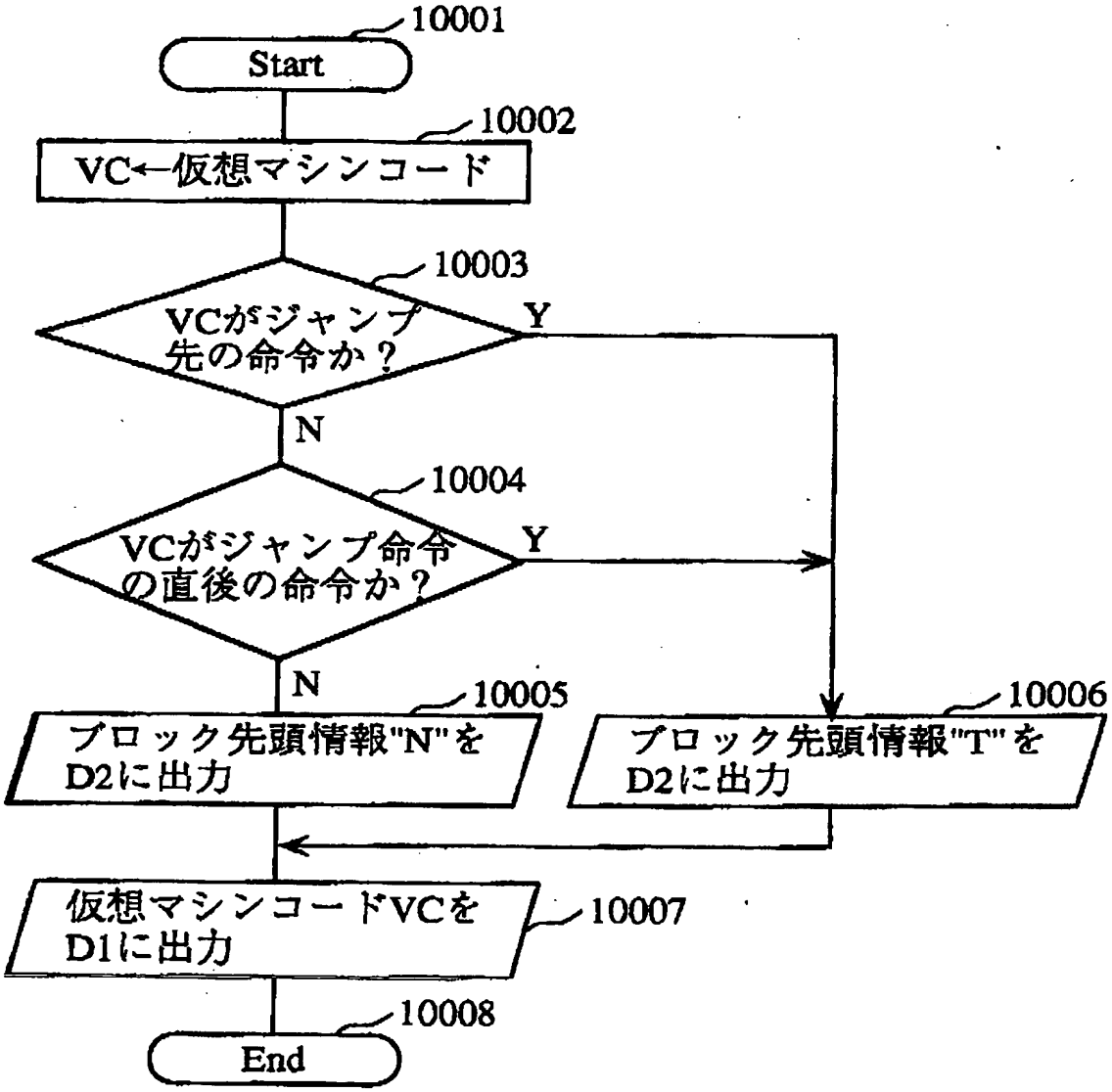
【図６５】


9701
Start
9702
変数bitsを空にする
9703
圧縮仮想マシンコード領域の
データを1ビット分読み込む
9704
変数bitsに読み込んだ
ビットデータを付加する
9705
展開情報領域を検索する
9706
該当するビット列が
見つかったか？
N
Y
9707
検索した展開情報から
仮想マシンコードと
オペランドを取得する
9708
VCaに展開した仮想
マシンコードを格納する
9709
配列op[ ]に展開した
オペランドを格納する
9710
End

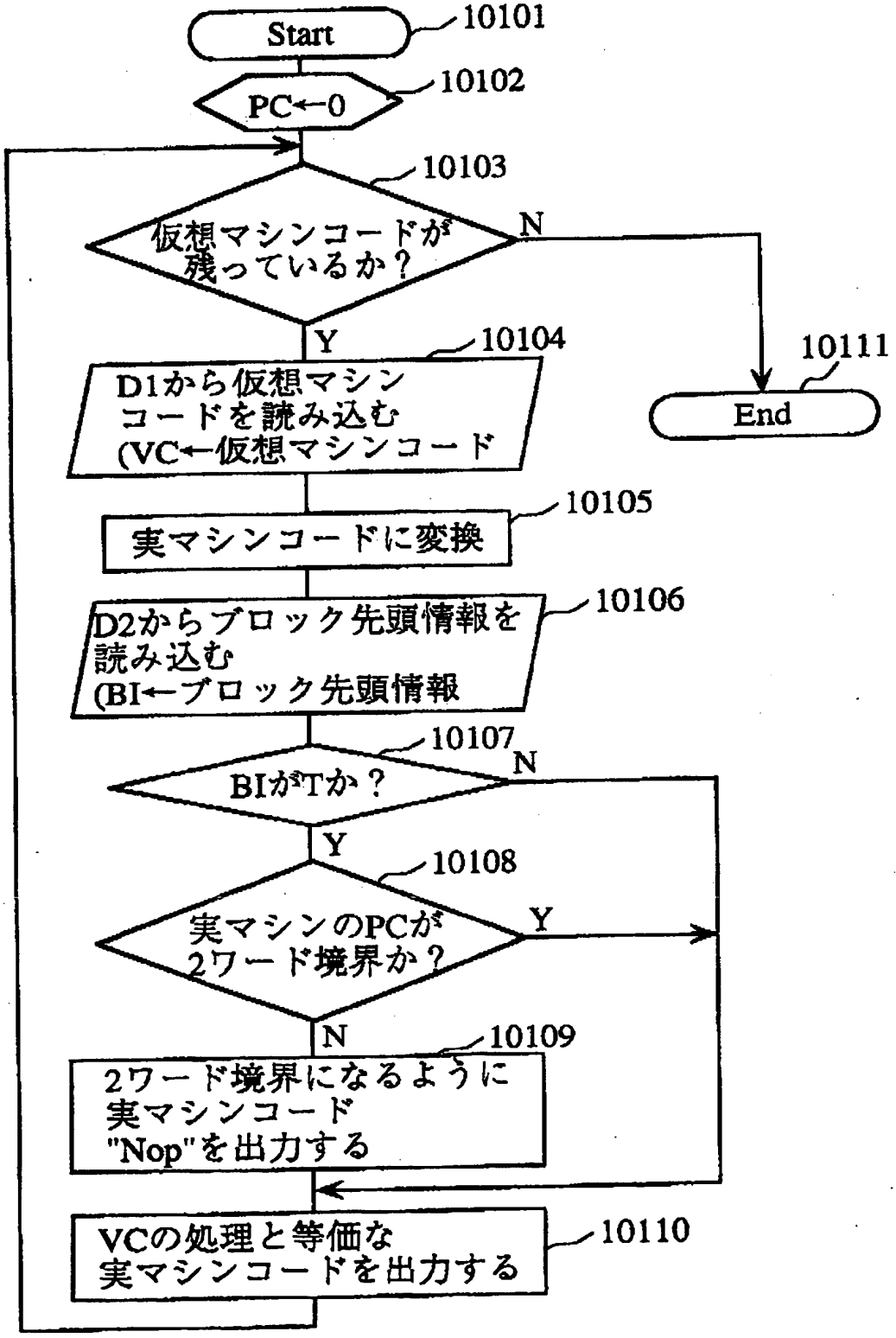
【図６６】


4310
ソースプログラム
4320
仮想マシン
コンパイラ
4321
出力部
ブロック先頭
情報生成部
4321a
D1
D2
4300
その場式
コンパイラ
4303
実マシンアドレス
保持部
4301
実マシン命令変換部
4302
分岐位置補正部
4311
実マシン
命令列

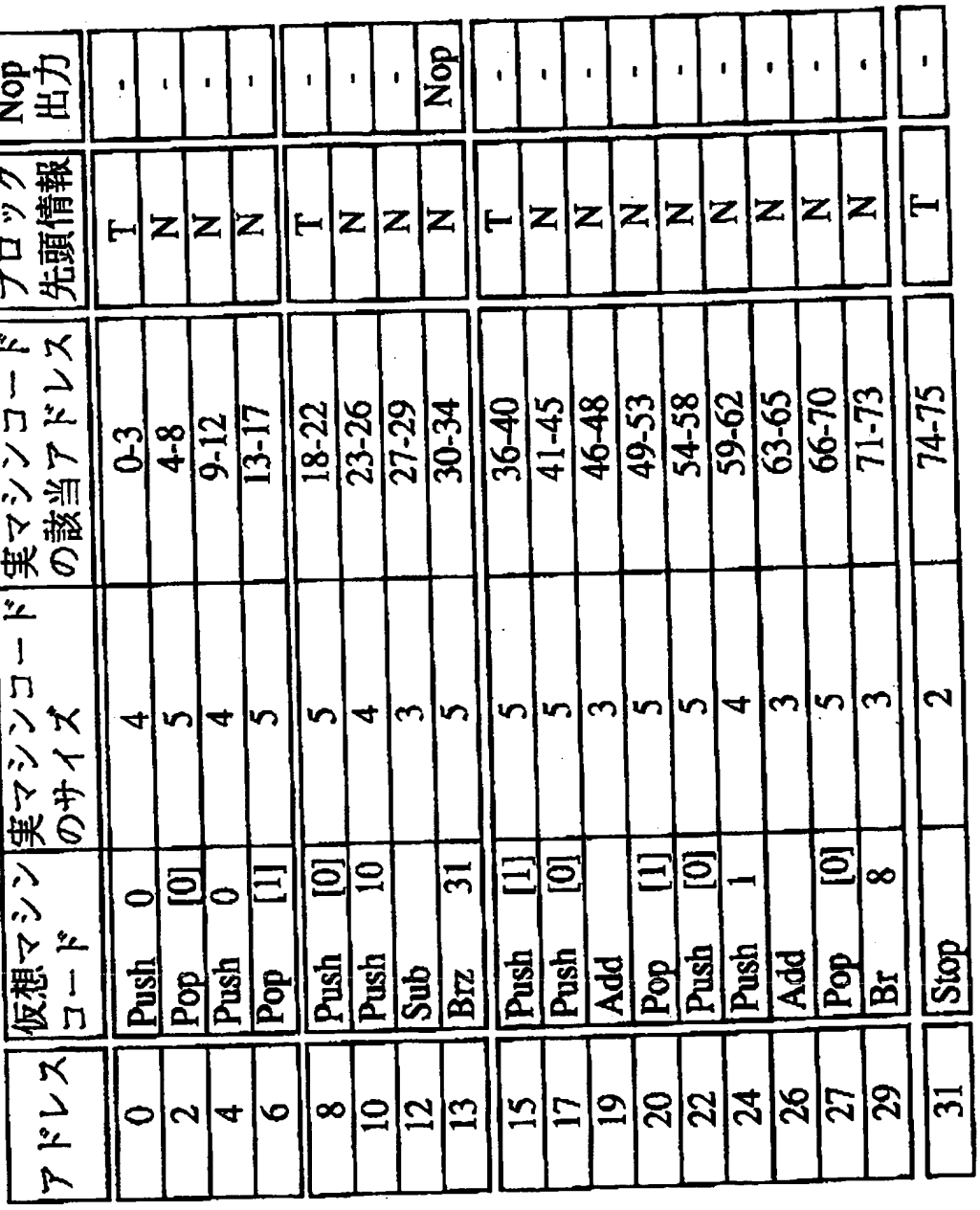
【図６７】


10001
Start
10002
VC←仮想マシンコード
10003
VCがジャンプ先の命令か？
Y
N
10004
VCがジャンプ命令の直後の命令か？
Y
N
10005
ブロック先頭情報"N"をD2に出力
10006
ブロック先頭情報"T"をD2に出力
仮想マシンコードVCをD1に出力
10007
10008
End

【図６８】

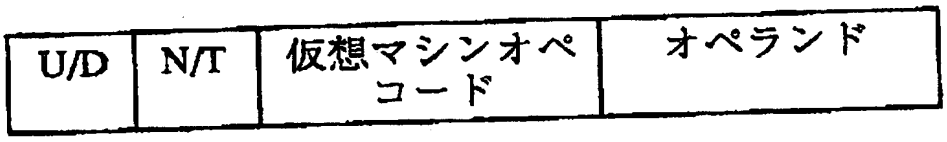

Start
10101
PC←0
10102
10103
仮想マシンコードが
残っているか？
N
Y
10104
D1から仮想マシン
コードを読み込む
(VC←仮想マシンコード
10111
End
実マシンコードに変換
10105
D2からブロック先頭情報を
読み込む
(BI←ブロック先頭情報
10106
10107
BIがTか？
N
Y
10108
実マシンのPCが
2ワード境界か？
Y
N
10109
2ワード境界になるように
実マシンコード
"Nop"を出力する
VCの処理と等価な
実マシンコードを出力する
10110

【図６９】

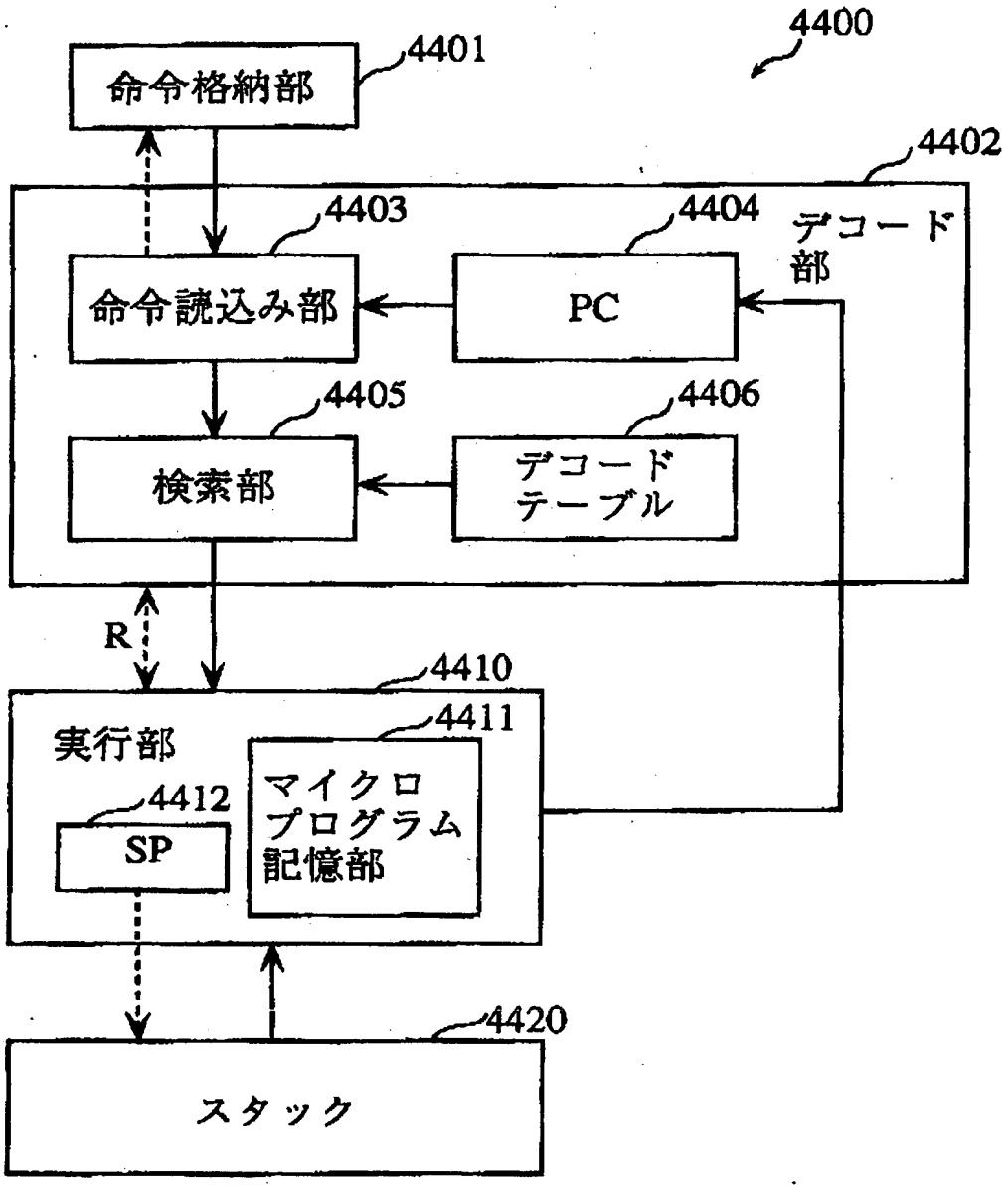


アドレス	仮想マシンコード	実マシンコードのサイズ	実マシンコードの該当アドレス	ブロック先頭情報	Nop出力
0	Push 0	4	0-3	T	-
2	Pop [0]	5	4-8	N	-
4	Push 0	4	9-12	N	-
6	Pop [1]	5	13-17	N	-
8	Push [0]	5	18-22	T	-
10	Push 10	4	23-26	N	-
12	Sub	3	27-29	N	-
13	Brz 31	5	30-34	N	Nop
15	Push [1]	5	36-40	T	-
17	Push [0]	5	41-45	N	-
19	Add	3	46-48	N	-
20	Pop [1]	5	49-53	N	-
22	Push [0]	5	54-58	N	-
24	Push 1	4	59-62	N	-
26	Add	3	63-65	N	-
27	Pop [0]	5	66-70	N	-
29	Br 8	3	71-73	N	-
31	Stop	2	74-75	T	-

【図７０】



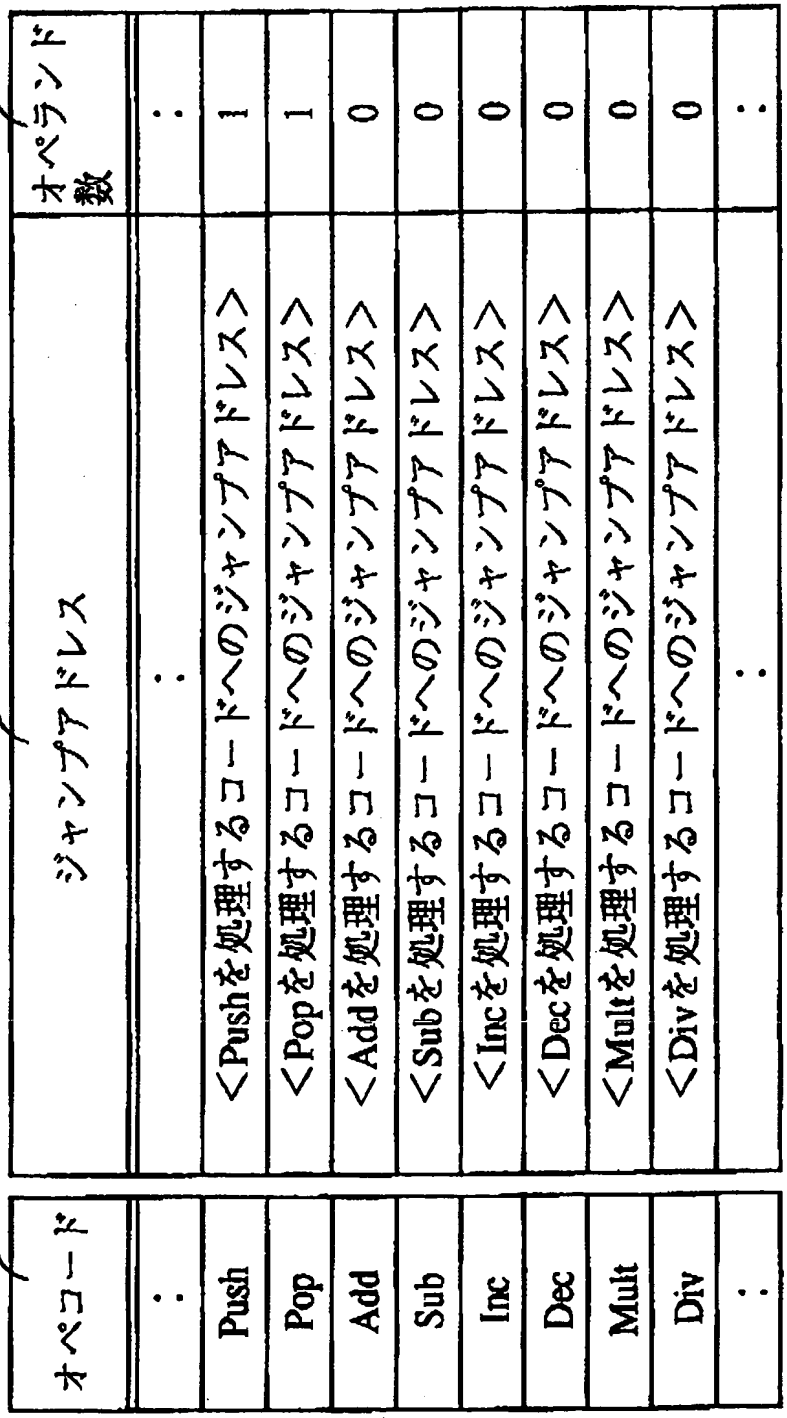
【図７１】


4400
4401
命令格納部
4402
4403
4404
デコード部
命令読込み部
PC
4405
4406
検索部
デコードテーブル
R
4410
実行部
4411
マイクロプログラム記憶部
4412
SP
4420
スタック

【図７２】

仮想マシン命令	意味	スタックの状態	スタックポインタの値
Push	オペランドをスタックにプッシュする	s0←operand	sp←sp+1
Pop	オペランドで指定されるアドレスに、スタックトップから二番目の値を格納する	operand←s0 s0←s1	sp←sp-1
Add	スタックトップの値にスタックトップから二番目の値を加え、結果をスタックトップに格納する	s0←s0+s1	sp←sp-1
Mult	スタックトップの値にスタックトップから二番目の値を掛け、結果をスタックトップに格納する	s0←s0*s1	sp←sp-1
Br	オペランドで指定されるアドレスに、無条件ジャンプする	s0←s0	sp←sp
Brz	スタックトップの値が0のとき、オペランドで指定されるアドレスにジャンプする	削除←s0 s0←s1	sp←sp-1
Brnz	スタックトップの値が0以外のとき、オペランドで指定されるアドレスにジャンプする	削除←s0 s0←s1	sp←sp-1
Call	オペランドで指定されるアドレスの関数を呼び出す	s0←次の仮想マシンコードアドレス	sp←sp+1
Ret	スタックトップの値のアドレスに、無条件ジャンプする	削除←s0 s0←s1	sp←sp-1
Stop	仮想マシンの実行を停止する	初期状態	sp←0

【図７３】


4406
4406a
4406b
4406c
オペコード
ジャンプアドレス
オペランド数
Push
Pop
Add
Sub
Inc
Dec
Mult
Div
<Pushを処理するコードへのジャンプアドレス>
<Popを処理するコードへのジャンプアドレス>
<Addを処理するコードへのジャンプアドレス>
<Subを処理するコードへのジャンプアドレス>
<Incを処理するコードへのジャンプアドレス>
<Decを処理するコードへのジャンプアドレス>
<Multを処理するコードへのジャンプアドレス>
<Divを処理するコードへのジャンプアドレス>
1
1
0
0
0
0
0
0

【図７４】

(a)

仮想マシン命令"Push"のマイクロプログラム		
1:Inc	r3	;仮想マシンのSPの値を1増やす
2:Load	r0,[r2]	;オペランドを取り出し、 ;レジスタ1番に読み込む
3:Inc	r2	;仮想マシンのPCを一つ増やし、 ;次の命令を読み込めるようにする
4:Store	[r3],r0	;レジスタ0番の内容を、 ;スタックに格納する
＜次の仮想マシン命令にジャンプするためのマイクロプログラム＞		

(b)

仮想マシン命令"Add"のマイクロプログラム		
1:Load	r0,[r3]	;スタックから値を取り出し、 ;レジスタ0番に読み込む
2:Dec	r3	;仮想マシンのSPの値を1減らす
3:Load	r1,[r3]	;スタックから値を取り出し、 ;レジスタ1番に読み込む
4:Add	r0,r0,r1	;レジスタ0番とレジスタ1番を加え ;結果をレジスタ1番に格納する
5:Store	[r3],r0	;レジスタ0番の内容を、 ;スタックに格納する
＜次の仮想マシン命令にジャンプするためのマイクロプログラム＞		

(c)

仮想マシン命令"Mult"のマイクロプログラム		
1:Load	r0,[r3]	;スタックから値を取り出し、 ;レジスタ0番に読み込む
2:Dec	r3	;仮想マシンのSPの値を1減らす
3:Load	r1,[r3]	;スタックから値を取り出し、 ;レジスタ1番に読み込む
4:Mult	r0,r0,r1	;レジスタ0番とレジスタ1番を掛け ;結果をレジスタ1番に格納する
5:Store	[r3],r0	;レジスタ0番の内容を、 ;スタックに格納する
＜次の仮想マシン命令にジャンプするためのマイクロプログラム＞		

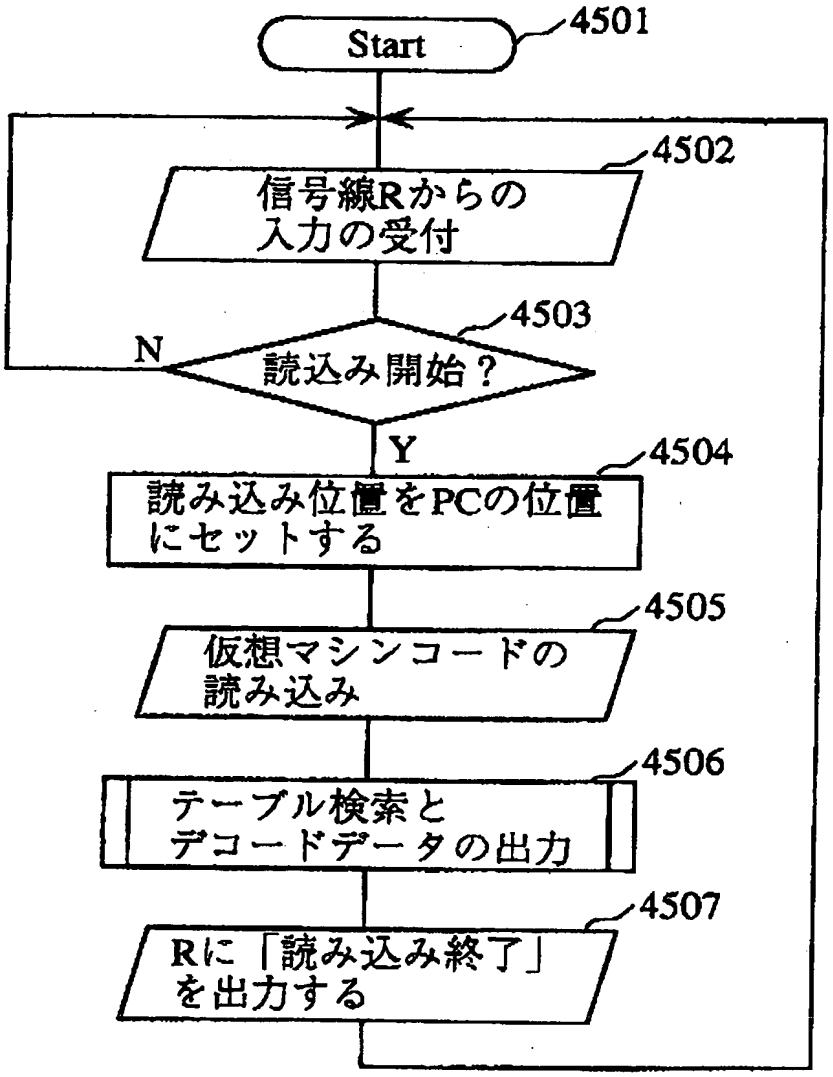
(d)

＜次の仮想マシン命令にジャンプするためのマイクロプログラム＞		
1:Load	r0,[r2]	;仮想マシンのPCの位置の仮想 ;マシン命令(ジャンプアドレス)を ;レジスタ0番に読み込む
2:Inc	r2	;仮想マシンのPCの値を1増やす
3:Jmp	r0	;レジスタ0番で示される位置に ;無条件ジャンプする

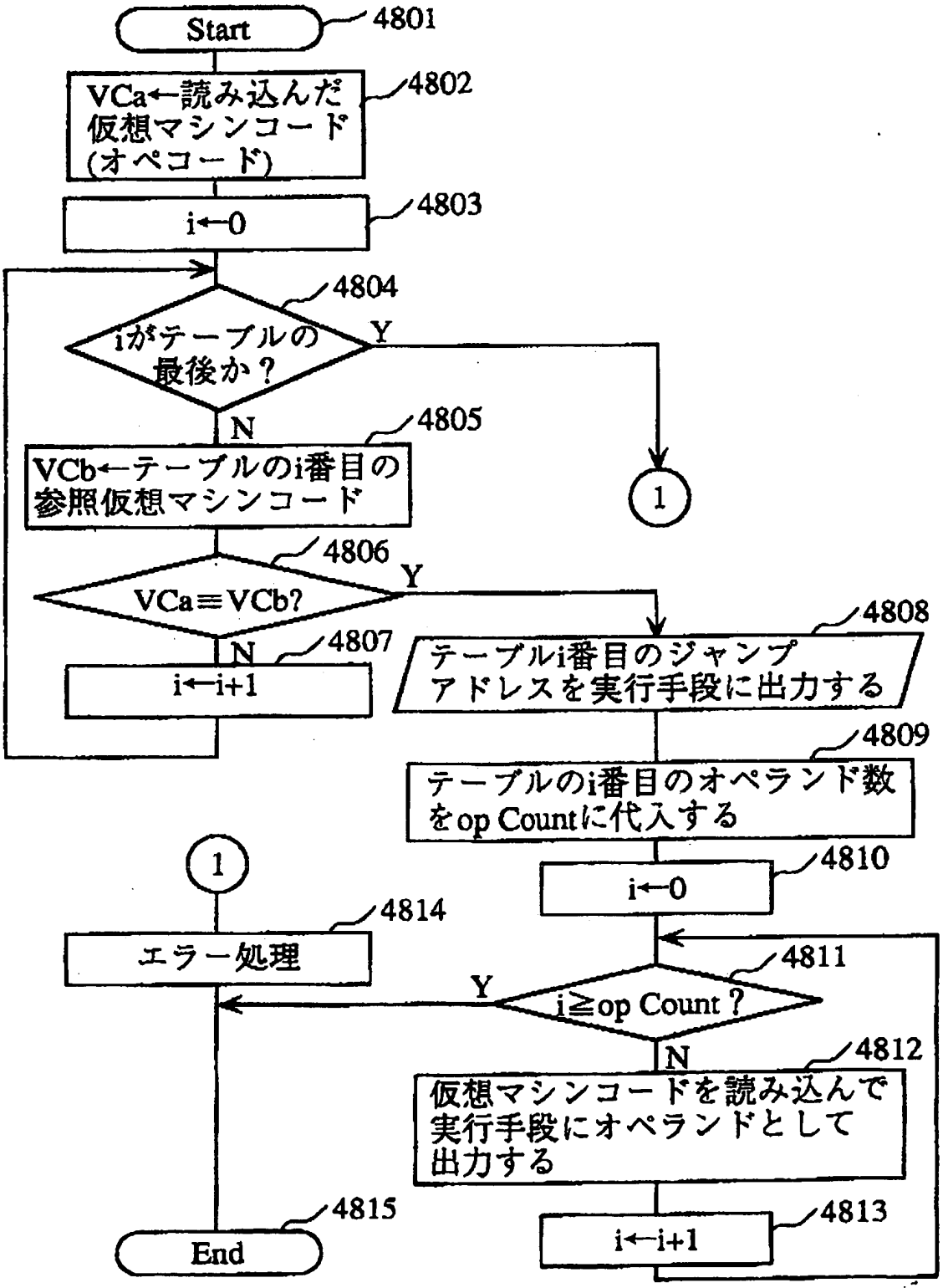
【図７５】

実マシン命令	意味	表記方法	例
Load	指定されたレジスタ(x0)の値に、指定されたレジスタの値(x1)か、レジスタで指定されたメモリ位置([x1])の値を格納する	Load x0,x1 Load x0,[x1]	Load r0,r1 Load r1,[r2]
Store	指定されたレジスタ(x1)の値を、レジスタで指定されたメモリ位置([x0])に格納する	Store [x0],x1	Store [r0],r1
Add	指定されたレジスタ(x1,x2)の値を加算し、値を指定されたレジスタ(x0)に格納する	Add x0,x1,x2	Add r0,r1,r2
Mult	指定されたレジスタ(x1,x2)の値を乗除し、値を指定されたレジスタ(x0)に格納する	Mult x0,x1,x2	Mult r0,r1,r2
Inc	指定されたレジスタ(x0)の値に1を加え、結果を同じレジスタに格納する	Inc x0	Inc r4
Dec	指定されたレジスタ(x0)の値に1を引き、結果を同じレジスタに格納する	Dec x0	Dec r3
Jmp	指定されたレジスタ(x0)の値のアドレスにジャンプする	Jmp x0	Jmp r2
Jz	指定されたレジスタ(x0)の値が0ならば、指定されたレジスタ(x1)のアドレスにジャンプする	Jz x0,x1	Jz r0,r2
Jnz	指定されたレジスタ(x0)の値が0以外ならば、指定されたレジスタ(x1)のアドレスにジャンプする	Jnz x0,x1	Jnz r0,r2
Nop	何もしない	Nop	Nop

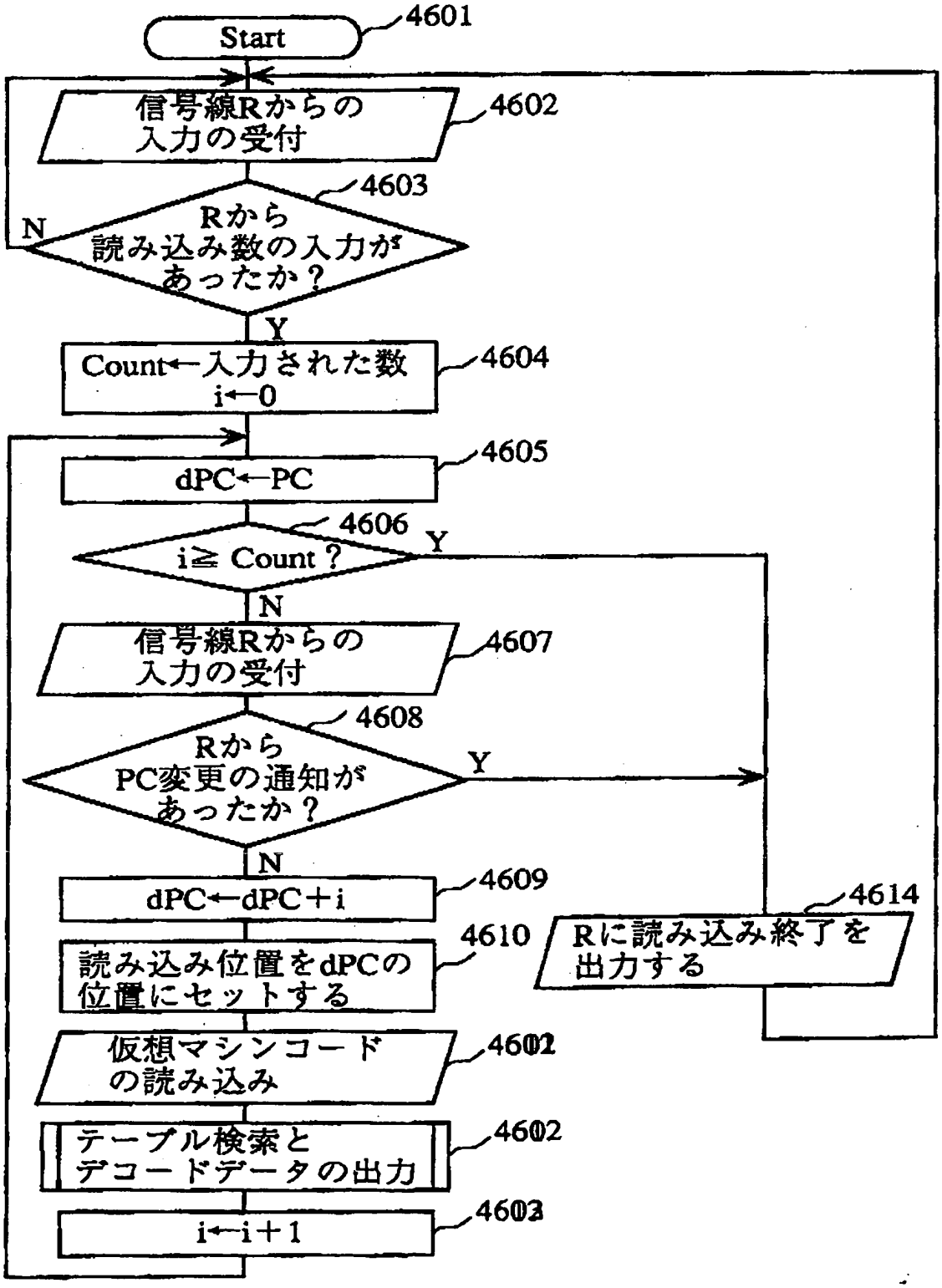
【図７６】


Start
4501
4502
信号線Rからの
入力の受付
4503
N
読込み開始？
Y
4504
読み込み位置をPCの位置
にセットする
4505
仮想マシンコードの
読み込み
4506
テーブル検索と
デコードデータの出力
4507
Rに「読み込み終了」
を出力する

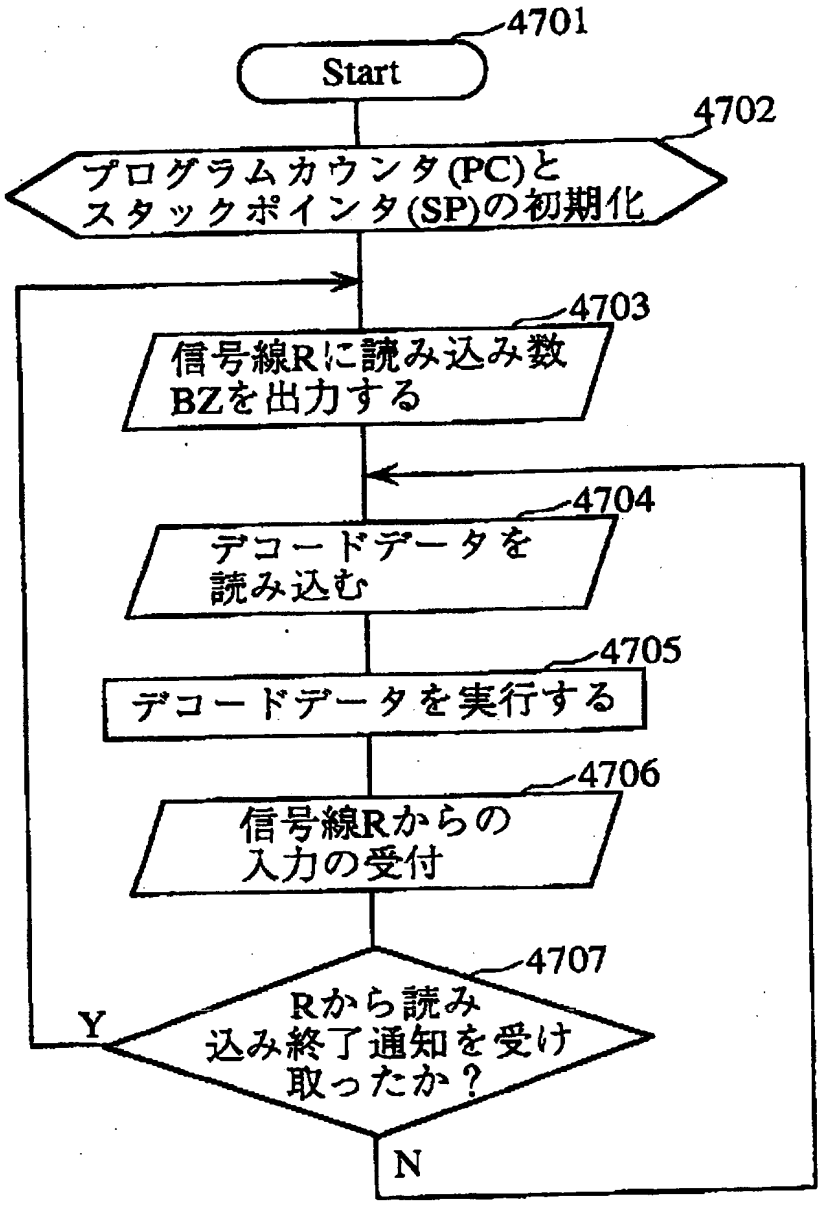
【図７７】


Start
4801
VCa←読み込んだ仮想マシンコード(オペコード)
4802
i←0
4803
4804
iがテーブルの最後か？
Y
N
4805
VCb←テーブルのi番目の参照仮想マシンコード
1
4806
VCa≡VCb?
Y
N
4807
i←i+1
4808
テーブルi番目のジャンプアドレスを実行手段に出力する
4809
テーブルのi番目のオペランド数をop Countに代入する
4810
i←0
1
4814
エラー処理
4811
i≧op Count？
Y
N
4812
仮想マシンコードを読み込んで実行手段にオペランドとして出力する
4813
i←i+1
4815
End

【図７８】


Start
4601
信号線Rからの
入力の受付
4602
4603
Rから
読み込み数の入力が
あったか？
N
Y
Count←入力された数
i←0
4604
dPC←PC
4605
4606
i≧ Count？
Y
N
信号線Rからの
入力の受付
4607
4608
Rから
PC変更の通知が
あったか？
Y
N
dPC←dPC＋i
4609
4610
読み込み位置をdPCの
位置にセットする
仮想マシンコード
の読み込み
テーブル検索と
デコードデータの出力
i←i＋1
4603
Rに読み込み終了を
出力する
4614

【図７９】


4701
Start
4702
プログラムカウンタ(PC)と
スタックポインタ(SP)の初期化
4703
信号線Rに読み込み数
BZを出力する
4704
デコードデータを
読み込む
4705
デコードデータを実行する
4706
信号線Rからの
入力の受付
4707
Rから読み
込み終了通知を受け
取ったか？
Y
N

【図８０】

(a)

1:	Push
2:	2
3:	Push
4:	3
5:	Push
6:	4
7:	Add
8:	Mult
9:	Pop
10:	0

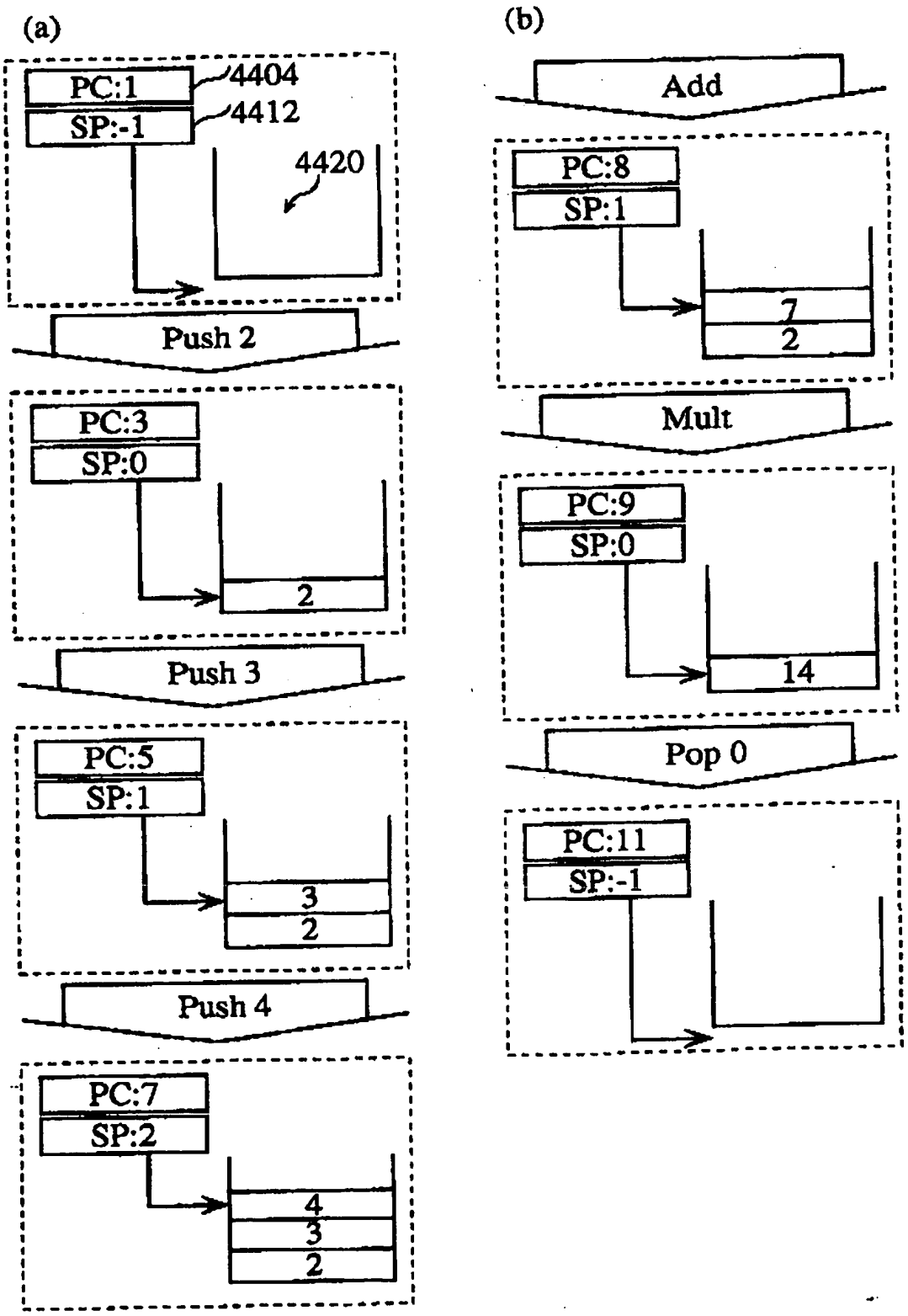
(b)

演算式：＜データ領域0番＞=2*(3+4)

(c)

1: ＜Pushを処理するコードへのジャンプアドレス＞
2: オペランド"2"
3: ＜Pushを処理するコードへのジャンプアドレス＞
4: オペランド"3"
5: ＜Pushを処理するコードへのジャンプアドレス＞
6: オペランド"4"
7: ＜Addを処理するコードへのジャンプアドレス＞
8: ＜Multを処理するコードへのジャンプアドレス＞
9: ＜Popを処理するコードへのジャンプアドレス＞
10: オペランド"0"

【図８１】


(a)
PC:1
SP:-1
4404
4412
4420
Push 2
PC:3
SP:0
2
Push 3
PC:5
SP:1
3
2
Push 4
PC:7
SP:2
4
3
2
(b)
Add
PC:8
SP:1
7
2
Mult
PC:9
SP:0
14
Pop 0
PC:11
SP:-1

【図８２】

(a)

仮想マシン命令"Push"のマイクロプログラム		
1:Inc	r3	;仮想マシンのSPの値を1増やす
2:Store	[r3],r0	;TOSレジスタ(0番)の値を ;スタックに格納する
3:Load	r0,[r2]	;オペランドを取り出し、 ;TOSレジスタに読み込む
4:Inc	r2	;仮想マシンのPCを一つ増やし、 ;次の命令を読み込めるようにする
＜次の仮想マシン命令にジャンプするためのマイクロプログラム＞		

(b)

仮想マシン命令"Add"のマイクロプログラム		
1:Load	r1,[r3]	;スタックから値を取り出し、 ;レジスタ1番に読み込む
2:Dec	r3	;仮想マシンのSPの値を1減らす
3:Add	r0,r0,r1	;レジスタ0番とレジスタ1番を加え、 ;結果をTOSレジスタに格納する
＜次の仮想マシン命令にジャンプするためのマイクロプログラム＞		

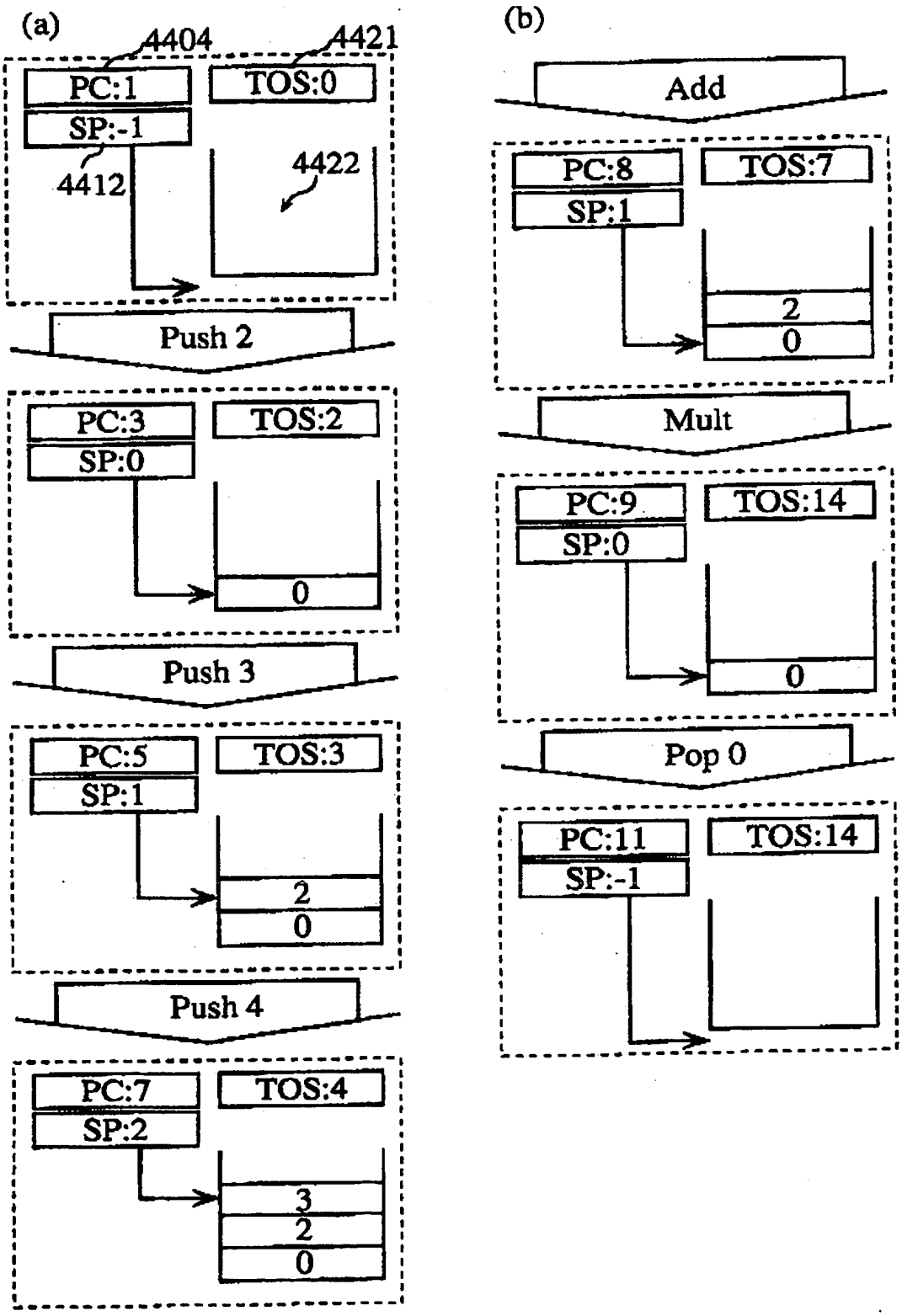
(c)

仮想マシン命令"Mult"のマイクロプログラム		
1:Load	r1,[r3]	;スタックから値を取り出し、 ;レジスタ1番に読み込む
2:Dec	r3	;仮想マシンのSPの値を1減らす
3:Mult	r0,r0,r1	;レジスタ0番とレジスタ1番を掛け ;結果をTOSレジスタに格納する
＜次の仮想マシン命令にジャンプするためのマイクロプログラム＞		

(d)

＜次の仮想マシン命令にジャンプするためのマイクロプログラム＞		
1:Load	r1,[r2]	;仮想マシンのPCの位置の仮想 ;マシン命令(ジャンプアドレス)を ;レジスタ1番に読み込む
2:Inc	r2	;仮想マシンのPCの値を1増やす
3:Jmp	r1	;レジスタ1番で示される位置に ;無条件ジャンプする

【図８３】


(a)
4404
4421
PC:1
TOS:0
SP:-1
4412
4422
Push 2
PC:3
TOS:2
SP:0
0
Push 3
PC:5
TOS:3
SP:1
2
0
Push 4
PC:7
TOS:4
SP:2
3
2
0
(b)
Add
PC:8
TOS:7
SP:1
2
0
Mult
PC:9
TOS:14
SP:0
0
Pop 0
PC:11
TOS:14
SP:-1

【図８４】

ステージ名	表記方法
インストラクションフェッチ	IF
デコード＆レジスタ参照	RF
実行	ALU
メモリ参照	MEM
レジスタ書き込み	WB

【図８５】

クロック	1	2	3	4	5	6	7	8
命令A	IF	RF	ALU	MEM	WB			
命令B		IF	RF	ALU	MEM	WB		
命令C			IF	RF	ALU	MEM	WB	
命令D				IF	RF	ALU	MEM	WB

【図８６】

クロック	1	2	3	4	5	6	7	8
命令A1	IF	RF	ALU	MEM	WB			
命令A2	IF	RF	ALU	MEM	WB			
命令B1		IF	RF	ALU	MEM	WB		
命令B2		IF	RF	ALU	MEM	WB		
命令C1			IF	RF	ALU	MEM	WB	
命令C2			IF	RF	ALU	MEM	WB	
命令D1				IF	RF	ALU	MEM	WB
命令D2				IF	RF	ALU	MEM	WB

【図８７】

クロック	1	2	3	4	5	6	7	8	9
命令A	IF	RF	ALU	MEM	WB				
命令B		IF	RF	・	ALU	MEM	WB		
命令C			IF	・	RF	ALU	MEM	WB	
命令D				・	IF	RF	ALU	MEM	WB

【図８８】

クロック	1	2	3	4	5	6	7	8	9
命令A1	IF	RF	ALU	MEM	WB				
命令A2	IF	RF	ALU	MEM	WB				
命令B1		IF	RF	・	ALU	MEM	WB		
命令B2		IF	RF	ALU	MEM	WB			
命令C1			IF	RF	ALU	MEM	WB		
命令C2			IF	RF	・	ALU	MEM	WB	
命令D1				IF	RF	ALU	MEM	WB	
命令D2				IF	RF	・	ALU	MEM	WB

【図８９】

クロック	1	2	3	4	5	6	7	8	9	10
命令A	IF	RF	ALU	MEM	WB					
命令B		IF	RF	・	・	ALU	MEM	WB		
命令C			IF	・	・	RF	ALU	MEM	WB	
命令D				・	・	IF	RF	ALU	MEM	WB

【図９０】

クロック	1	2	3	4	5	6	7	8	9
命令A1	IF	RF	ALU	MEM	WB				
命令A2	IF	RF	ALU	MEM	WB				
命令B1		IF	RF	・	・	ALU	MEM	WB	
命令B2		IF	RF	ALU	MEM	WB			
命令C1			IF	RF	ALU	MEM	WB		
命令C2			IF	RF	ALU	MEM	WB		
命令D1				IF	RF	ALU	MEM	WB	
命令D2				IF	RF	・	ALU	MEM	WB

【図９１】

クロック	1	2	3	4	5	6	7
命令A	IF	RF	ALU	MEM	WB		
命令B		IF	x				
命令C			IF	RF	ALU	MEM	WB

【図９２】

クロック	1	2	3	4	5	6	7
命令A1	IF	RF	ALU	MEM	WB		
命令A2	IF	RF	ALU	MEM	WB		
命令B1		IF	x				
命令B2		IF	x				
命令C1			IF	RF	ALU	MEM	WB
命令C2			IF	RF	ALU	MEM	WB

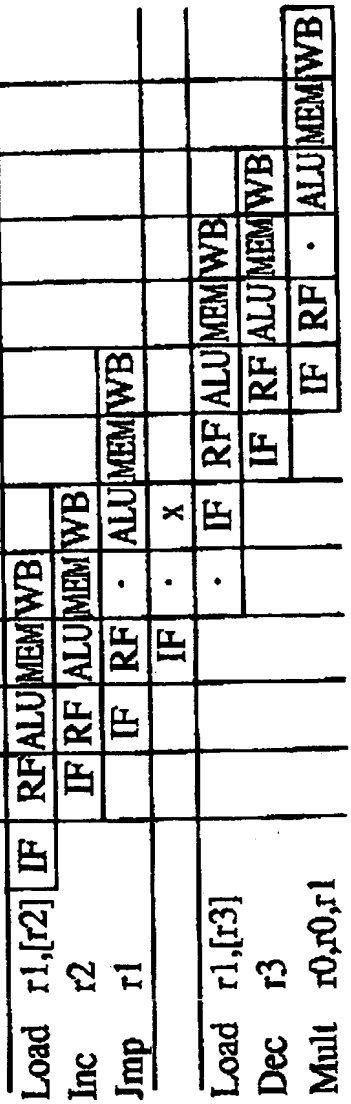
【図９３】

クロック	1	2	3	4	5	6	7	8	9	10	11
Load r1,[r2]	IF	RF	ALU	MEM	WB						
Inc r2		IF	RF	ALU	MEM	WB					
Jmp r1			IF	RF	ALU	MEM	WB				
				IF	x						
Load r1,[r3]					IF	RF	ALU	MEM	WB		
Dec r3						IF	RF	ALU	MEM	WB	
Mult r0,r0,r1							IF	RF	ALU	MEM	WB

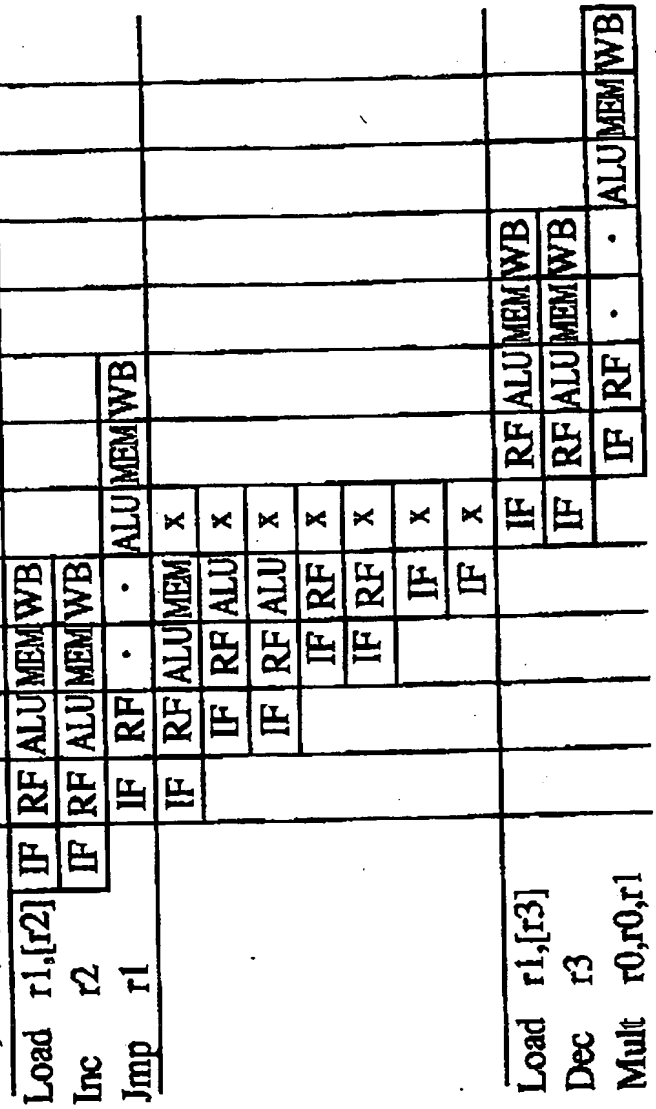
【図９４】

クロック	1	2	3	4	5	6	7	8	9	10	11
Load r1,[r2]	IF	RF	ALU	MEM	WB						
Inc r2	IF	RF	ALU	MEM	WB						
Jmp r1		IF	RF	・	ALU	MEM	WB				
		IF	RF	ALU	x						
			IF	RF	x						
			IF	RF	x						
				IF	x						
				IF	x						
Load r1,[r3]					IF	RF	ALU	MEM	WB		
Dec r3					IF	RF	ALU	MEM	WB		
Mult r0,r0,r1						IF	RF	・	ALU	MEM	WB

【図９５】


クロック 1 2 3 4 5 6 7 8 9 10 11 12 13
Load r1,[r2] IF RF ALU MEM WB
Inc r2 IF RF ALU MEM WB
Jmp r1 IF RF ALU MEM WB
IF RF ALU MEM WB
x
Load r1,[r3] IF RF ALU MEM WB
Dec r3 IF RF ALU MEM WB
Mull r0,r0,r1 IF RF ALU MEM WB

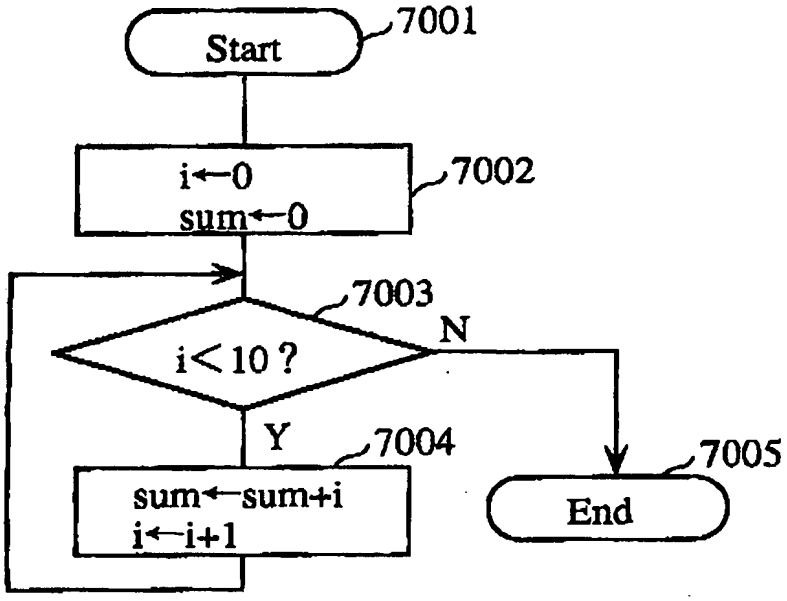
【図９６】


クロック 1 2 3 4 5 6 7 8 9 10 11 12 13
Load r1,[r2] IF RF ALU MEM WB
Inc r2 IF RF ALU MEM WB
Jmp r1 IF RF ALU ・ ALU MEM WB
IF RF ALU MEM x
IF RF ALU x
IF RF ALU x
IF RF x
IF RF x
IF x
x
Load r1,[r3] IF RF ALU MEM WB
Dec r3 IF RF ALU MEM WB
Mult r0,r0,r1 IF RF ・ ・ ALU MEM WB

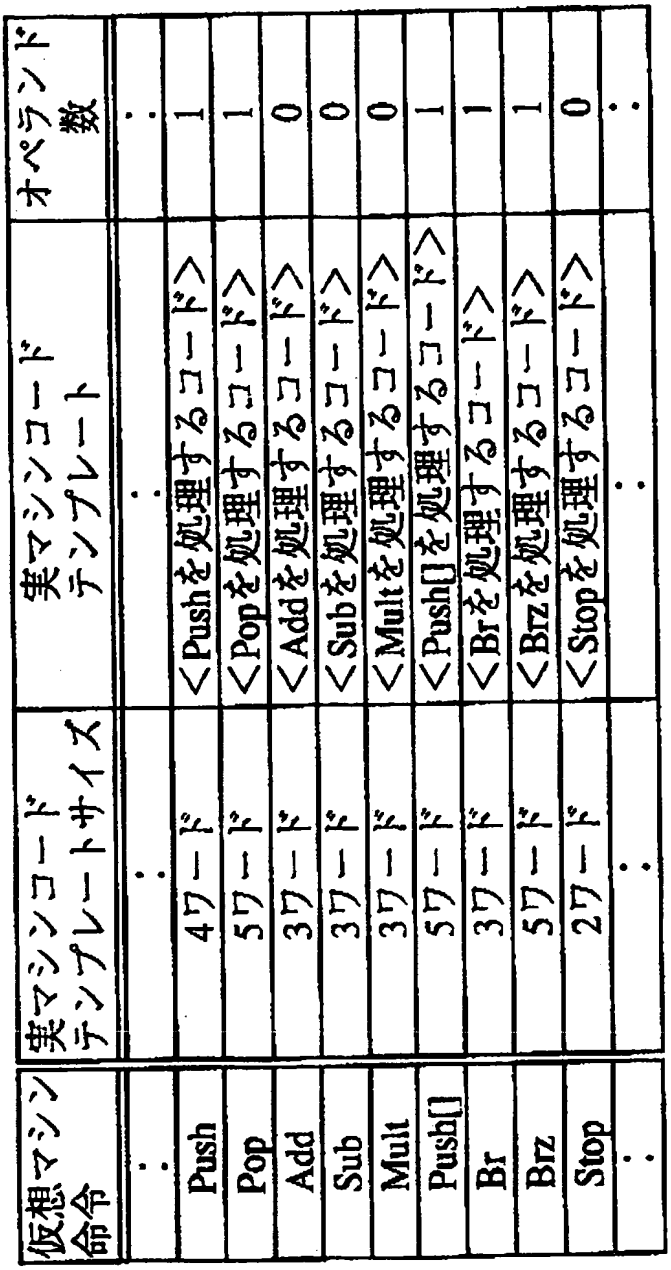
【図９７】

```
 0:Push   0      ;i←0
 2:Pop    [0]
 4:Push   0      ;sum←0
 6:Pop    [1]

 8:Push   [0]    ;i＜10？
10:Push   10
12:Sub
13:Brz    31

15:Push   [1]    ;sum←sum+i
17:Push   [0]
19:Add
20:Pop    [1]
22:Push   [0]    ;i←i+1
24:Push   1
26:Add
27:Pop    [0]
29:Br     8

31:Stop
```

【図９８】



【図９９】



仮想マシン命令	実マシンコードテンプレートサイズ	実マシンコードテンプレート	オペランド数
・・	・・	・・	・・
Push	4ワード	<Pushを処理するコード>	1
Pop	5ワード	<Popを処理するコード>	1
Add	3ワード	<Addを処理するコード>	0
Sub	3ワード	<Subを処理するコード>	0
Mult	3ワード	<Multを処理するコード>	0
Push[]	5ワード	<Push[]を処理するコード>	1
Br	3ワード	<Brを処理するコード>	1
Brz	5ワード	<Brzを処理するコード>	1
Stop	2ワード	<Stopを処理するコード>	0
・・	・・	・・	・・

【図１００】

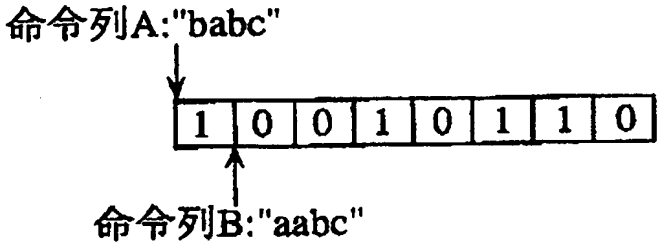
仮想マシンコードのアドレス	仮想マシンコード	実マシンコードのサイズ	実マシンコードの該当アドレス
0	Push 0	4	0-3
2	Pop [0]	5	4-8
4	Push 0	4	9-12
6	Pop [1]	5	13-17
8	Push [0]	5	18-22
10	Push 10	4	23-26
12	Sub	3	27-29
13	Brz 31	5	30-34
15	Push [1]	5	35-39
17	Push [0]	5	40-44
19	Add	3	45-47
20	Pop [1]	5	48-52
22	Push [0]	5	53-57
24	Push 1	4	58-61
26	Add	3	62-64
27	Pop [0]	5	65-69
29	Br 8	3	70-72
31	Stop	2	73-75

【図１０１】

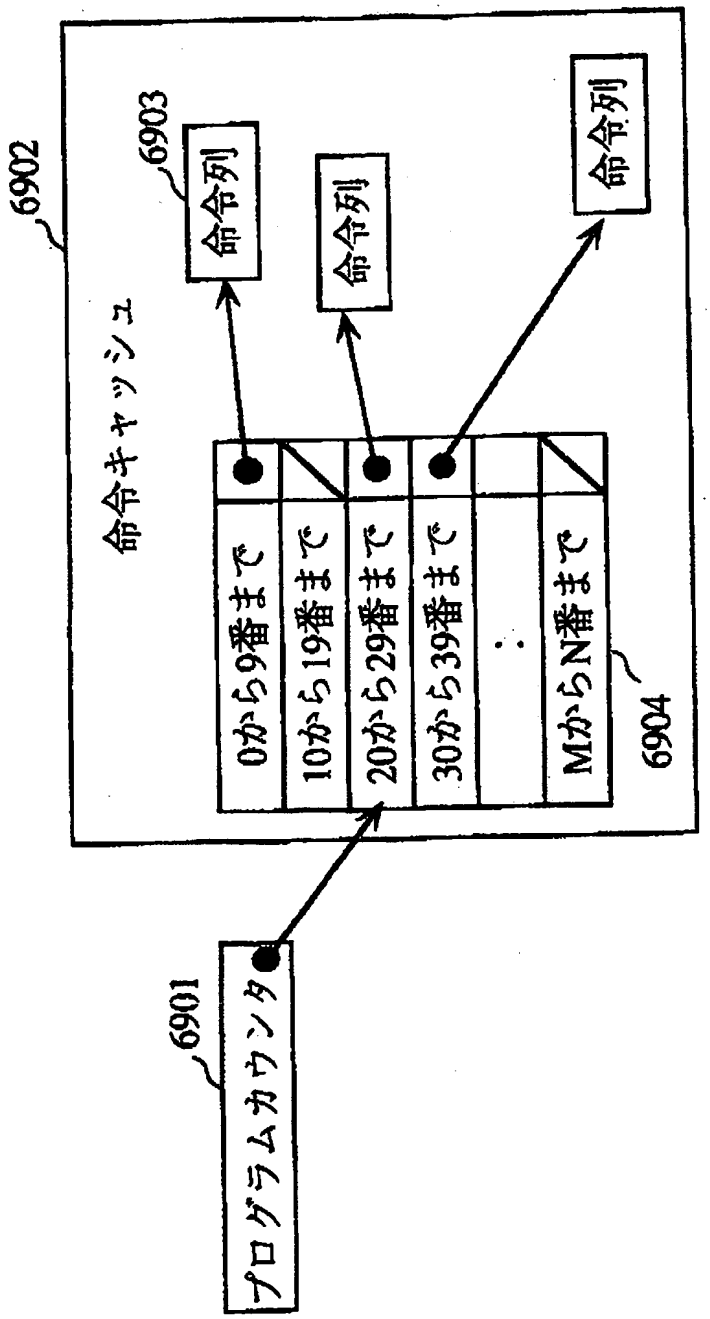
(a)

ビット列 (9300a)	意味 (9300b)
0	"a"
10	"b"
110	"c"
111	"d"

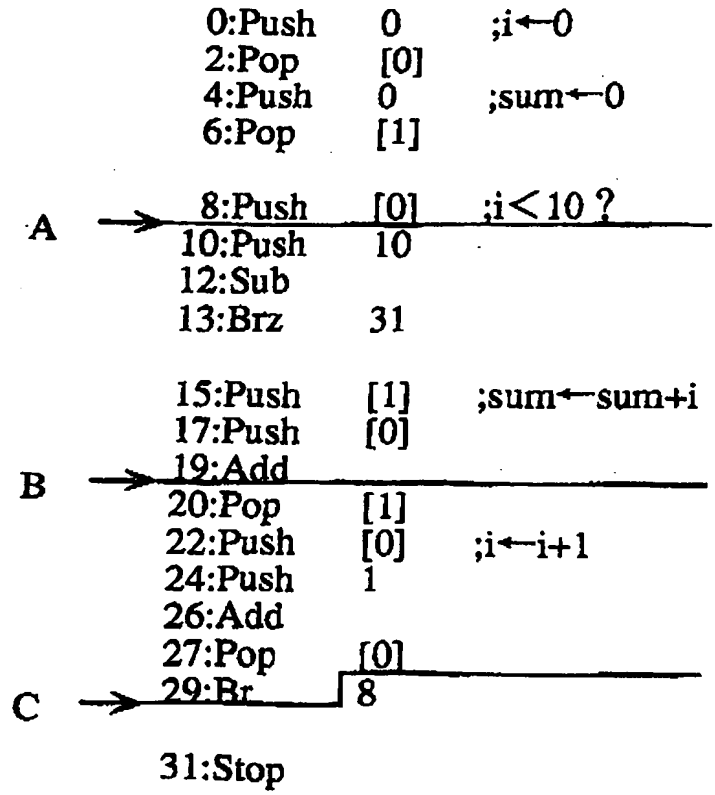
(b)



【図１０２】


6902
命令キャッシュ
6903
命令列
命令列
命令列
0から9番まで
10から19番まで
20から29番まで
30から39番まで
MからN番まで
6904
プログラムカウンタ
6901

【図１０３】


0:Push 0 ;i←0
2:Pop [0]
4:Push 0 ;sum←0
6:Pop [1]
A
8:Push [0] ;i＜10？
10:Push 10
12:Sub
13:Brz 31
15:Push [1] ;sum←sum+i
17:Push [0]
B
19:Add
20:Pop [1]
22:Push [0] ;i←i+1
24:Push 1
26:Add
27:Pop [0]
C
29:Br 8
31:Stop

【書類名】　　要約書

【要約】

【課題】　データ依存関係を吸収することにより高速実行を可能とした仮想マシンを提供する。

【解決手段】　スタック型仮想マシンであって、最上段TOSと次段SOSが実マシン２０１のレジスタに割り当てられたスタック１２０と、実行対象となる仮想マシン命令列が格納された命令格納部１０２と、それら仮想マシン命令それぞれに対応づけられた情報であって、対応づけられた仮想マシン命令に後続する仮想マシン命令が実行された場合のスタック１２０での記憶段数が増加するか減少するかを示す先行命令情報が格納された先行命令情報格納部１０１と、それらから次に実行すべき仮想マシン命令及び対応する先行命令情報を読み出して解読するデコード部１０３と、解読された仮想マシン命令の演算処理と先行命令情報に基づくスタック１２０の先行処理とを実行する実行部１１０とを備える。

【選択図】　　図２

【書類名】　職権訂正データ
【訂正書類】　特許願

＜認定情報・付加情報＞
【特許出願人】
【識別番号】　000005821
【住所又は居所】　大阪府門真市大字門真１００６番地
【氏名又は名称】　松下電器産業株式会社
【代理人】　申請人
【識別番号】　100090446
【住所又は居所】　大阪市北区豊崎３丁目２番１号　淀川５番館６Ｆ
中島国際特許事務所
【氏名又は名称】　中島　司朗

出　願　人　履　歴　情　報

識別番号　　〔０００００５８２１〕

１．変更年月日　　１９９０年　８月２８日
［変更理由］　新規登録
住　所　　大阪府門真市大字門真１００６番地
氏　名　　松下電器産業株式会社